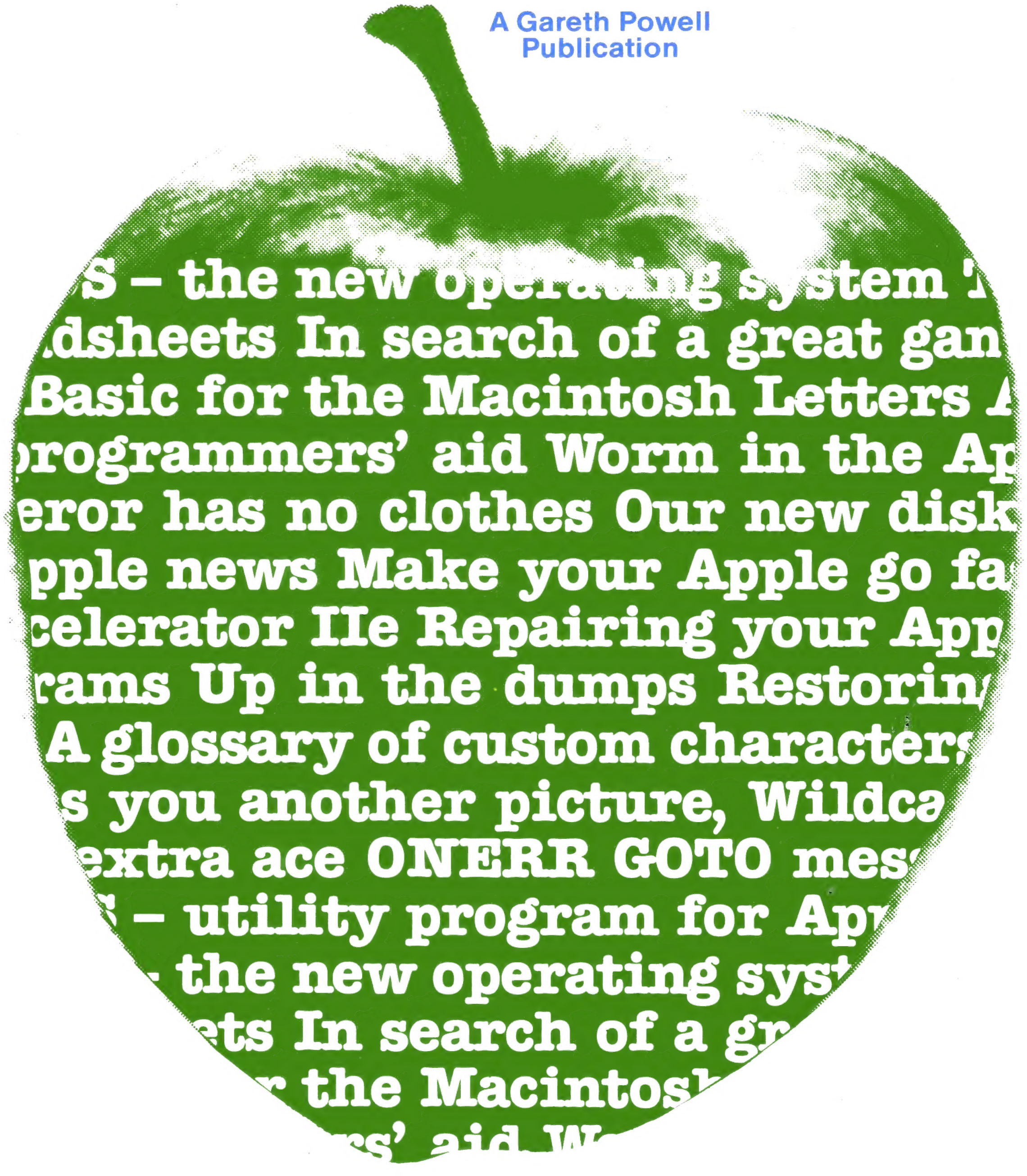
The Independent Magazine

Vol 1 No 7 August $3*

The Australian Apple Review

A Gareth Powell Publication


S – the new operating system
dsheets In search of a great gan
Basic for the Macintosh Letters
programmers' aid Worm in the Ap
eror has no clothes Our new disk
pple news Make your Apple go fa
celerator IIe Repairing your App
rams Up in the dumps Restorin
A glossary of custom characters
s you another picture, Wildca
extra ace ONERR GOTO mes
– utility program for Ap
the new operating syst
ets In search of a gr
the Macintos
rs' aid W

Registered by Australia Post Publication No. NBG6313

*Recommended retail price

Applicard

a new generation in CP/M™ for your Apple™ II, II+, //e, or ///

- Shift key modification
- Optional counter/timer chip
- 64k &/or 128k RAM extender
- 64k or 128k RAM disk under CP/M
- 64k, 128k or 192k RAM disk under Apple DOS
- Supports most Apple peripherals
- Up to 90k printer buffer (with 80-col extended board)
- Fast //e video driver
- //e programmable function keys
- 6MHz Z-80B processor
- 64k RAM on-board
- 2k EPROM (expandable)
- CP/M 2.2 + utilities
- 70 Column Hi-Res capabilities
- 80-column video board support
- Interface for networking and communications
- SAMS CP/M primer
- Full installation manual
- OEM documentation available
- Apple /// Applicard comes with WORDSTAR free & has many features utilizing the unique features of the Apple ///.
- Support of CORVUS, RANA & other mass storage devices

• DEALER ENQUIRIES WELCOME •

alphaSyntauri

The computer music system that enhances your ability to:

☐ CREATE
☐ LEARN
☐ IMPROVISE
☐ PLAY
☐ EDUCATE
☐ EXPLORE
☐ COMPOSE

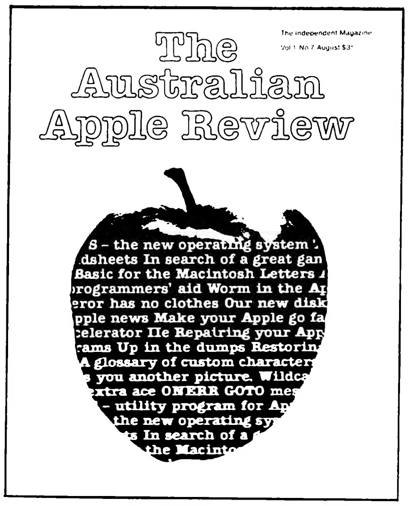
COMPUMUSIC

PO Box 57 – 103 St. Johns Rd,
Glebe, NSW 2037
692 9293 PHONE 692 9875

Macintosh programs coming soon

The Australian Apple Review

Contents


The independent Magazine
Vol 1 No 7 August $3*
The Australian Apple Review
S - the new operating system
dsheets In search of a great gan
Basic for the Macintosh Letters
rogrammers' aid Worm in the A
eror has no clothes Our new disk
pple news Make your Apple go fa
celerator IIe Repairing your App
rams Up in the dumps Restorin
A glossary of custom character
you another picture. Wildca
xtra ace ONERR GOTO mes
- utility program for Ap
the new operating sy
s In search of a
the Macinto

The Australian Apple Review
Top Rear, 4 Carrington Rd,
Randwick, NSW 2031
Phone: (02) 398 5111

Published by Gareth Powell Pty. Ltd.

Editor: Gareth Powell
Art Director: Roy Bisson
Production Manager: Sian Powell
Production Assistant: Gary Williams
Photographer: Gary Williams
Subscription Manager:
Tina Spathos

Typesetting:
Printaction Pty. Ltd.

Distribution:
Network, SYDNEY

Printed by:
Ian Liddel Commercial Printing

This magazine is designed by either Roy Bisson or Avrille Williams, depending on who is on holiday, who is the busiest and so on. Both have strong, indeed robust, mental constitutions. Both are being reduced to nervous whimpering wrecks by having to bromide very faint copies of programs sent in by contributors. Some printouts are so faint that they can only be deciphered under a north light with a strong magnifying glass.

If you are sending in a copy of a program, would you please use a newish ribbon in your printer. As far as we can ascertain, most programmers start with the original ribbon in their printer and hang on to it through the years without ever investing in another.

I know the feeling. I used to keep my ribbon until the printing turned into grey on grey in grey. Then I tried to recycle our ribbons in a series of weird and wonderful ways – none of which worked, but they left my hands permanently dyed. Now I buy my ribbons through Contact Computer Supplies – nice people to deal with – and change them at regular intervals.

If you are going to send us in a program please, please, make sure that the printing is legible and fairly dark. Otherwise we have the devil's own job reproducing it. Thank you.

Following on with that theme, we have now closed our program competition because we had difficulty in working out a fair system of judging the very few entries we received. Taking the coward's way out we have closed the contest.

This does not mean that we want you to stop sending in programs that you feel will be of interest to other readers. Simply that there is no

contest involved anymore. Life is enough of a contest, anyway.

On a separate note, please observe that the new Apple disk magazine is being launched next month. As it is unprotected it may be that you'll be making copies for your friends. That's perfectly OK. This disk magazine is more of a service to Apple users than a money making proposition. An order form appears elsewhere in this magazine. □

MEMOREX

THE DISC FOR EVERY APPLE

8" 5.25" 3.5"

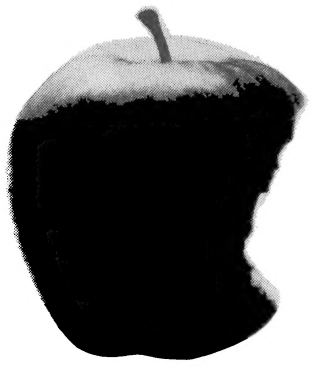
5 YEAR WARRANTY

Syd. (02) 908-2211
Melb. (03) 267-2955
Adel. (08) 79-9951
Bris. (07) 376-5677, (07) 52-7333
Perth. (09) 446-6100
Cnbra. (062) 81-1977

Remember: For your Data . . .

'When it matters make it Memorex'

Apple news



The CMOS 6502

The Apple IIc has made a fairly major impact on the market. Partially it is the design. Partially it is the machine's inherent compatibility.

And part of it is the new microprocessor which drives the machine. It comes from Rockwell, is designated 65CØ2 and is the CMOS (that stands for Complementary Metal-Oxide Semiconductor) version of the standard 65Ø2.

The chip offers various advantages, such as low power consumption and lack of heat output. It is software compatible with the standard chip but also contains some functions which fill in a number of gaps in the original 65Ø2's instruction set.

The standard 65Ø2 microprocessor probably had as much to do with the computer revolution as any other microprocessing chip in the world.

It had a simple, straight-forward instruction set and simple interfacing requirements. It worked extremely well in comparison with other eight-bit processors such as the Z-8Ø but it did have restrictions on the use of certain addressing modes and there were a few minor anomolies.

Better instructions

For example, on the 65Ø2 several instructions did not behave quite the way the documentation suggested. One of the main problems with the 65Ø2 was the branch instruction timing. The problem lay, in the main, with the documentation. Although it was correctly specified on the original data sheets, several tutorials incorrectly stated how long a branch instruction took.

Nearly all of these glitches in the 65Ø2 have been corrected in the 65CØ2 and already there is a library of literature building around this chip.

Several of the instructions built-in are quite new, and the 65CØ2 also enables some existing addressing modes to be used with instructions which did not accept those modes on the original 65Ø2.

The 65CØ2 has the electrical characteristics you would expect from a CMOS integrated circuit in that it uses minute amounts of power and therefore creates little or no heat.

Six times the speed

Most importantly, the chip can be made much faster in operation with relative ease. Versions for speeds up to six megahertz will probably be available sometime in the not-too-distant future.

The standard Apple 65Ø2 worked at one megahertz, so such a speed increase is quite dramatic.

Bear in mind the IBM PC works at 4.77 megahertz and you can see what a difference this speed increase will make.

The chip's consumption is also extremely low which will make battery-powered operation feasible when this chip is combined with the new CMOS memory chip.

Nevertheless, its most impressive feature is that it .is almost totally compatible with 65Ø2 specifications. This permits all of Apple hardware and software to be used with the new processor. The 65CØ2 represents a major step forward in Apple technology.

College education at home

In the United States it is now possible to take your college courses at home using a computer-telephone modem link-up.

The most advanced stage is at the New Jersey Institute of Technology. They have spent a fortune and a large amount of time to set up an electronic information exchange system which offers 14 complete continuing educational seminars.

The system was developed by the Professor of Computer Science at the New Jersey Institute of Technology, Murray Turoff, who has been working on the system since 1976. The system went on line in 1983 with a very short series of continuing educational seminars.

This has attracted the attention of educators and institutions around the world. Participation of students is not restricted to the New Jersey area.

Differences

The lessons conducted on this computer telecommunications system are very different from those offered in the conventional classroom setting.

The first big advantage is that limitations of time and location virtually do not exist.

You can access your computer at any time of the day or night.

You can take as long or short lessons as you need to.

Doctor Turoff points out with justifiable pride that the offering of courses via the computer is very flexible. It allows people whose business commitments would otherwise prevent them extending their education, to join a seminar group. He says one of the major areas where they have attracted students is in the middle management level. Executives are taking the opportunity for professional enrichment courses.

Another major difference is that a computer classroom has a group far more widespread than a conventional classroom.

The NJIT has set up a seminar conferencing system which allows students to see each other's answers to questions and remarks in discussion.

In the normal classroom once a question has been answered that is the end of it.

Not in this electronic environment.

Each student works at home to get an answer to the question and then keys it in. Only after it has been transmitted can the sender see what the other students have said and compare answers.

This system encourages participation and stimulates curiosity.

It has another big advantage. It allows anonymous answers.

Students who are shy about putting up their hands in class – and that is most of us – often feel more comfortable working in a computer setting. A group member who wants to try an unusual or outrageous idea may do so without fear of embarrassment.

Unlimited communication

The system allows unlimited communication between each of the participants and the instructors. As soon as a user logs on he checks to see if there is any mail waiting for him.

One of the great attractions of the system is that members can schedule their class times to suit their own needs.

There is no actual need for a member to be sitting at his terminal at a specific time, although the system is capable of supporting online conferences, if they want to discuss a specific subject at a specific time.

Whenever the student accesses the system the current lesson prepared by the instructor is waiting. A member may find other members of the group have already responded to the topic and the instructor has passed comments, all of which can help in the educational process.

If the student is still living in the dark ages and prefers hard copy it can be dumped onto the printer.

Sending in lessons requires a word processing program on the student's computer. The copy can then be transmitted in the normal way through a modem.

No tests, no quizzes

In this particular system there are no tests or quizzes as the seminar course progresses and no grades are given at the end. This means the user has to be extremely disciplined and willing to use free time in working on the educational material. This may be fairly easy in the case of a mature executive who has invested serious money in taking part in it. The same may not be true of younger students. Some may duck courses and fall behind, some will stay current with the work.

It may be some sort of mark encouragement is going to be needed before the system will work properly with young students.

There is a charge for this New Jersey system. Tuition for one three month course is $525 with additional fees for text and modem connect charges. The only other outlay is for a modem and a communications package.

It is going to be very interesting to see when the idea will arrive in Australia. Of course, in a modified form, it has been here for many, many years indeed.

It has been the basis of most of the bush radio schools. This is merely the modern extension. □

Worm in the Apple

Flash your Mac

I am, as everyone knows, not a prude. I do not even support Fred Nile. But even I look a little askance when I see that Apple is running Macintosh education schemes. And that the walls of the classrooom are covered with graffiti, put up by Apple staff, which reads, "Flash your Mac", "Cuddle a Mouse" and "Orwell had one". Not, I think, in the best of taste.

The Macintosh is named, so I am assured, after a variety of apple. Not after dirty old men who flash, most of whom, I am quite sure, are computer programmers between jobs. All I can say is thank heavens this incident didn't happen in Australia.

Rodent plague

If there is justice Apple management are about to get their knuckles severely rapped.

They have now introduced the abominable mouse to the Apple IIe, which until that moment had been a lovely machine of which I was particularly fond.

Is nothing sacred? □

Letters

Additions to mailing program (Vol 1, No 5)

Dear Sir,

For the mailing of journals registered in Category B, Australia Post requires that they should be presented in bundles assembled in numerical order of Post Code (PC) within a number of sorting divisions. Due to the rather eccentric arrangement of post codes with divisions, a simple sort of addresses in numerical order of post code does not suffice. For example, the following table shows the divisions of Sydney south of the harbour (ND = 4) – has been done by computer. This is particularly irritating if the job is being done by volunteers!

The enclosed additions to Birthisel's program cause the post code of a new entry to the list to be "tagged" by prefixing it with a CTRL character. This will not be printed but will be sorted in order of its ASCII number. The resulting sort thus provides a list in which all the CTRL–A members appear first, arranged in numerical order of post code, followed by the CTRL – B members and so on.

The DATA in line 3ØØØ are appropriate to the Divisions given above, the first item being the number of Divisions (ND = 4), and each group of post codes being preceded by the number of post code ranges within the division (NZ).

Sorting Division	Post code range	Program Tag	NZ
Sydney City	2ØØØ – 2ØØ5	CTRL – A	1
Rushcutters Bay	2Ø1Ø – 2Ø11	CTRL – B	2
Mail Centre	2Ø21 – 2Ø3Ø		
Southwest suburbs	2127 – 2144	CTRL – C	3
	216Ø – 22Ø2		
	2555 – 2574		
Southern suburbs	2ØØ6 – 2ØØ9	CTRL – D	4
	2Ø12 – 2Ø2Ø		
	2Ø31 – 2Ø59		
	22Ø3 – 2249		

If there is a large number of journals to be mailed, sorting manually is time consuming, even if an initial sort into numerical order

The CTRL characters G, H, M and U have a special function and cannot be used so they must be excluded if the number of Divisions on the list is large, and line 152Ø has been modified because a string which starts with a CTRL cannot be displayed on the screen.

A program for "tagging" an existing list is easily written.

```
1Ø32 PC =  VAL (PC$): READ ND: FOR I = 1 TO ND: READ NZ: FOR J = 1 TO NZ
1Ø34  READ C1: READ C2: IF PC < C1 OR PC > C2 GOTO 1Ø38
1Ø36 PC$ =  CHR$ (I) + PC$: GOTO 1Ø39
1Ø38  NEXT J: NEXT I
1Ø39  RESTORE
152Ø  PRINT A2$(I); SPC( 3); RIGHT$ (PC$(I),4)
1732  PRINT : IF  LEFT$ (T$,1) = "Y" THEN  PRINT "FOR ALPHABETICAL ORDER";
      TAB( 35);"TYPE 1": PRINT : PRINT "FOR POST CODE ORDER"; TAB( 35);"TY
     PE 2": GET LO$: GOSUB 27ØØ: GOTO 175Ø
1738  HOME : GOTO 172Ø
275Ø  FOR J = I TO 1 STEP  - D: IF LO$ = "1" GOTO 276Ø
2752  IF PC$(J) <  = PC$(J + D) GOTO 283Ø
2754  GOTO 277Ø
3ØØØ  DATA  4,1,2ØØØ,2ØØ5,2,2Ø1Ø,2Ø11,2Ø21,2Ø3Ø,3,2127,2144,216Ø,22Ø2,2555
     ,2574,4,2ØØ6,2ØØ9,2Ø12,2Ø2Ø,2Ø31,2Ø59,22Ø3,2249
```

E D Poppleton
Hunters Hill, NSW

Graham Black's "help" call

Dear Graham,

Regarding your "help" call related to the underscore cursor. I'm not into 8Ø column cards on IIe, but perhaps the pokes to location $3ØF mentioned on page 43 of March 84 Softalk will help.

bytes
E9 4Ø at $3ØF nonflashing
A9 AØ at $3ØF invisible
A9 DF at $3ØF underline

These are effective when the following routine is used as an input exit.

3ØØ: A9 OB A2 Ø3 85 38 86 39
3Ø8: 4C EA Ø3 48 31 28 38 E9
31Ø: 4Ø 91 28 68 3C 21 FD

I seem to have this out of sequence, but you may see what is intended (1. I wish I had a photocopier; 2. I should have used the word processor. Ah, well.)

How did I track this down? Using "Daryl's Apple Digest", of course.

Daryl Jones
Malvern, Vic.

Ed: *"Daryl's Apple Digest" searches the major Apple computer magazines and records all articles and items relevant to Apple II users. For more information, apply to 26 Parslow St, Malvern, Vic 3144.* ■

Puzzle program

Dear Sir,

I found your June edition very interesting and informative and I am including with this letter my cheque for 12 months subscription.

I noticed the program submitted by Greg Bailey (the Lotto selector program) that was printed in the June edition. In case you are looking for more such programs to include in the programming section of the magazine, I am submitting a small puzzle program with this letter.

The program simulates one of those little 4 x 4 square puzzles you used to find in showbags, containing 15 tiles and one blank space. You have to shuffle the tiles around to make words or number patterns. Actually, I saw such a program as this working on a Macintosh recently, and so I wrote this one for my Apple.

Errol Chopping
Dubbo, NSW

The program presents a 4 x 4 grid containing 15 characters and one blank space. The aim is to shuffle the characters about, trying to make words or number patterns.

1	2	3	4
5	6	7	8
9	Ø	A	B
C	D	E	

Only those characters adjacent to the blank space may be moved and they are moved into the blank space. To move the cursor, use the key I (up), J (left), K (right), M (down) and to push a character into the space use the space bar.

You may exit the program at any time by pressing ESC.

The constants defined in the subroutine at line 1ØØØ are as follows:

CX	horizontal position of top left corner
CY	vertical position of top left corner
X,Y	grid co-ordinates of current cursor position
D	distance between characters on grid
W	length of word entered
KB	address of keyboard buffer
SB	keyboard strobe
BX,BY	co-ordinates of blank space
PX,PY	screen co-ordinates of cursor position
P (4,4)	array containing status of each cell. (=Ø not allowed to move, =1 allowed to move)

The routine at line 2ØØØ allows you to enter the characters you want in the puzzle. These are held in the string W$. Line 2Ø15 pads out W$ with random characters if you enter less than 15. The loop at 2Ø3Ø-2Ø5Ø prints the characters of W$ on the screen and substitutes the hyphen (-) for any blanks you may have entered.

The main part of the program, from line 4Ø to line 28Ø, works like this:
LINE 5Ø collects the ASCII code from the screen and converts it into the string A$. Note, memory locations 4Ø and 41 hold a pointer to the address of the current vtab line and location 36 holds the horizontal position of the cursor. So the statement
VTAB PY:HTAB PX:P=PEEK(4Ø)+256*PEEK(41)+PEEK(36)
will set P equal to the memory location of the screen position VTAB PY:HTAB PX.
LINE 7Ø looks at the keyboard to see what key you are pressing and when one is pressed clears the keyboard strobe and sets K equal to its ASCII value.
LINES 8Ø and 9Ø adjust the position of the cursor according to a keystroke of I, J, K or M.
LINE 2ØØ sends contol back to line 4Ø if you try to push a character into the space from an illegal position.
LINES 21Ø-23Ø swap the chosen character with the blank space.
LINES 24Ø-27Ø adjust the status array P(X,Y) so that only those cells adjacent to the blank space may be legally moved according to the new grid pattern.

```
]LIST

10  REM
--------------
TEXT PUZZLE
E.G.CHOPPING
09.07.84
--------------   .
20  GOSUB 1000: REM  (CONSTANTS)
30  GOSUB 2000: REM  (GET LETTERS)
40 X = (PX - CX) / D:Y = (PY - CY) / D
50  VTAB PY: HTAB PX:P =  PEEK (40) + 256 *  PEEK (41) +  PEEK (36):P =  PEEK
     (P) - 128:A$ =  CHR$ (P)
60  INVERSE : PRINT A$;: NORMAL
70 K =  PEEK (KB): ON (K < 127) GOTO 70:K = K - 128: POKE SB,0: ON (K = 27
    ) GOTO 3000: ON (K = 32) GOTO 200: VTAB PY: HTAB PX: PRINT A$;
80 PX = PX + D * (K = 75) * (X < 4) - D * (K = 74) * (X > 1)
90 PY = PY + D * (K = 77) * (Y < 4) - D * (K = 73) * (Y > 1)
100  GOTO 40
200  IF  NOT P(X,Y) THEN 70
210  VTAB PY: HTAB PX: PRINT " ";
220  VTAB BY: HTAB BX: PRINT A$;
230 BX = PX:BY = PY
240  FOR I = 1 TO 4: FOR J = 1 TO 4:P(I,J) = 0: NEXT J,I
250 P(X,Y - (Y > 1)) = 1:P(X,Y + (Y < 4)) = 1
260 P(X - (X > 1),Y) = 1:P(X + (X < 4),Y) = 1
270 P(X,Y) = 0
280  GOTO 40
290  REM
---------------------------------
DEFINE CONSTANTS

1000  DIM P(4,4)
1010 CX = 20:CY = 5:X = 4:Y = 4:D = 3:W = 0:KB =  - 16384:SB =  - 16368
1020 BX = X * D + CX:BY = Y * D + CY:PX = BX:PY = BY
1030 P(4,3) = 1:P(3,4) = 1
1050  RETURN : REM
---------------------------------
GET CHARCTERS FOR PUZZLE

2000  TEXT : HOME : SPEED= 255: NORMAL : NOTRACE : VTAB 18: PRINT "ENTER U
     P TO 15 CHARACTERS (NO SPACES)": VTAB 20: HTAB 11: PRINT "AND PRESS '
     RETURN'": VTAB 22: HTAB 12: FOR I = 1 TO 15: PRINT  CHR$ (95);: NEXT
     : VTAB 22: HTAB 12
2010  INPUT "";W$
2015  IF  LEN (W$) < 15 THEN W$ = W$ +  CHR$ ( INT ( RND (1) * 26) + 64): GOTO
     2015
2017 W$ =  LEFT$ (W$,15) + " "
2020  HOME : VTAB 20: PRINT " 'ESC' TO EXIT ": PRINT "'SPACE' TO PUSH": PRINT
     "I,J,K,M TO MOVE": NORMAL
2030  FOR I = 1 TO 4: FOR J = 1 TO 4:W = W + 1: VTAB I * D + CY: HTAB J *
     D + CX: IF  MID$ (W$,W,1) = " " THEN  PRINT "-";: GOTO 2050
2040  PRINT  MID$ (W$,W,1);
2050  NEXT J,I
2060  VTAB Y * D + CY: HTAB X * D + CX: PRINT " ";
2070  RETURN : REM
---------------------------------
CONFIRM ENDING

3000  VTAB 1: HTAB 1: PRINT "ANOTHER GAME (Y/N)?";: GET Z$: PRINT Z$;: IF
     Z$ = "Y" THEN  RUN
3005  VTAB 2: HTAB 1: PRINT "WANT TO QUIT (Y/N)?";: GET Z$: PRINT Z$;: IF
     Z$ = "Y" THEN  VTAB 23: END
3010  IF Z$ = "N" THEN  VTAB 1: HTAB 1: CALL  - 868: VTAB 2: HTAB 1: CALL
     - 868: GOTO 70
3020  GOTO 3000
```

When you can't afford mistakes.

There really is a difference in diskette brands.

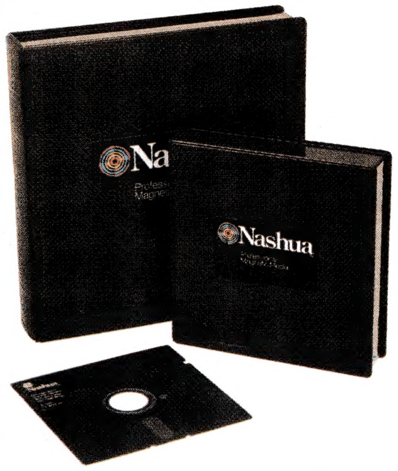
It's in the way they are made.

Because the computer industry cannot afford variation in the quality of diskettes, at Nashua we looked for, and found a way to ensure absolute consistency.

Here's how we do it.

Quality Circles.

At Nashua we've found the best way to attain this 'consistency' is to ensure that at each stage of production our diskettes are statistically checked to make sure the quality is 'built-in' every step of the way.



Rather than long production lines, we have 'Quality Circles' – small groups of people whose job it is to make sure that each Nashua diskette is right in the first place.

The result is a diskette with such consistency, that it is chosen by those people who can't afford mistakes.

Phone Sydney 958 2044, Canberra 47 0511, Melbourne 428 0501, Brisbane 369 4244, Adelaide 42 0021, Perth 328 1888, Darwin 81 6204, Auckland 77 5468, Wellington 85 1965, Christchurch/Dunedin Whitcoulls Ltd, PNG 25 6766.

Nashua

There really is a difference.

Ruwald & Skinner 402 NA/A

Our new disk magazine

We have been looking at the possibility of launching an Apple disk magazine. To do this we got hold of some of the disk magazines which are available in the United States.

We expected to see a screen version of what appears in the text with a few programs thrown in.

Not at all. The ones we looked at are not translations of paper magazines into floppy disk format but are intelligent and original entities with their own editorial approaches. There are several currently available in the United States.

Impressive

The one which impressed us most was Mentor which is intended for users of PCs. This was not the first disk magazine on the market. That honour goes to IB Magazette (a horrid title) which came out in August 1982. The IB Magazette in its introduction section says "IB Magazette will play several valuable roles: software supply, magazine, user group, amusement, trouble shooter, forum."

In fact, the function it handles best is as an unofficial user group. The magazine costs $15.

Everything on the disk is available for the reader to copy, use, keep in a software library. And this is the great advantage of disk magazines.

While any computer program can be listed in a printed magazine, it is the devil's own job to type it in carefully and correctly.

With a disk magazine you can copy directly onto another disk. Interestingly, IB Magazette accepts no advertising whatsoever.

Mentor

Mentor first came out in 1983, almost a year after the IB Magazette. It has taken a slightly different road.

Mentor uses professional writers and provides program enhancers to upgrade popular software such as WordStar, dBase II and Lotus 1-2-3. Mentor also accepts advertising.

The advertisements are listed in the main menu so if you want to see an advertisement, you must call it up from the main menu. These advertisements are in themselves fascinating, because they frequently contain teasers (an interactive demonstration of a program) to show you the sort of thing for sale. Rather similar to the trailers they are now putting on video tapes.

Mentor obviously has disk space problems but sometimes program demonstrations for new products are included on separate disks.

Mentor also gets round the problem of disk space by issuing you with a master disk which contains the operating system. Putting the operating system on a separate disk makes Mentor only slightly less user-friendly but leaves far more space for lengthier programs and articles.

The future

We believe there is a future for this sort of magazine in Australia.

So that is what we are going to do.

In August we will be releasing the first of our new monthly disk magazines. The price will be $1Ø including postage which we think is an extremely fair price, all things being taken into consideration. It makes it, as far as we can ascertain, the cheapest disk magazine in the world.

It will contain programs, subroutines, advertisements, a minimum of editorial, practical tips and suggestions. As well, it will occasionally be accompanied by another disk which will be manufacturer's offerings, so you can test before you buy.

If you would like to be in on the genesis of a new magazine please send your cheque or money order (payable to **The Australian Apple Review**) to **Top Rear, 4 Carrington Rd, Randwick 2Ø31**. Include your name, address, age and the number of months you wish to subscribe for. There is no minimum, so it is quite acceptable to send only $1Ø for the first month. □

Trends in spreadsheets

by Duncan McCann

Early spreadsheets were fairly simple programs. They let a user put information into a worksheet and manipulate it to a degree. The first was VisiCalc, and whether it was that amazing piece of software that started the personal computer revolution or the Apple machine it ran on, is now immaterial.

What is important, is that in the spreadsheet the personal computer found a reason for being, a justification for its existence.

From those early beginnings we have come a long way with fairly sophisticated products.

Trends

The following trends appear to have evolved:

1) Integration of increasingly complex functions into the software.

You can get anything ranging from linear programming to calculus depending on your needs and the program that you use.

2) The ability to read information from files created by other programs.

Many programs allow you to read out the information to another program such as a word processor. Many of the programs allow you to access information from programs such as databases. Both of these attributes increase the flexibility of the spreadsheet tremendously.

3) The integration of other related programs to the spreadsheet as exemplified by Lotus 1-2-3 and Framework.

This is an area that will be expanding tremendously in the future. So far Lotus 1-2-3 is available on the Macintosh but it will certainly be available for the Apple IIe early next year. The follow-up program, Symphony, will come some months later. Note well that several other software houses already have similar programs in the pipeline and this area will be the scene of a ferocious sales battle between the major companies. Only the consumer can benefit.

4) A movement towards the standardisation of data formats.

More and more we are finding spreadsheet programs are using the same sort of information entry styles, so moving from one spreadsheet to another does not mean you have to start at square one and re-learn everything.

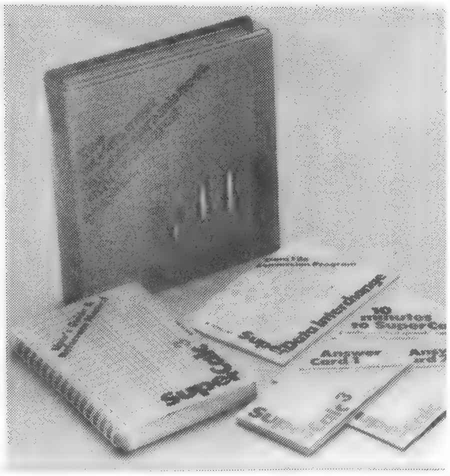
5) The use of macro commands to allow the user to set up spread sheets to a predetermined formula with the minimum of effort.

Writing these macros, as they are called, is not easy and the skills required are somewhat akin to programming. But the difference they make in setting up a new spreadsheet is measured in hours of work, not minutes. No doubt we will see simpler versions of macros available in the near future.

Still evaluating

Integrated packages are still, of course, being evaluated. But there is no doubt programs such as MBA, Lotus 1-2-3, Symphony and Framework are the way of the future because spreadsheets do not exist in a vacuum. They are part of a total business process.

As a result they need access to data bases.



They need access to word processing.

They need to be used with a graphing facility.

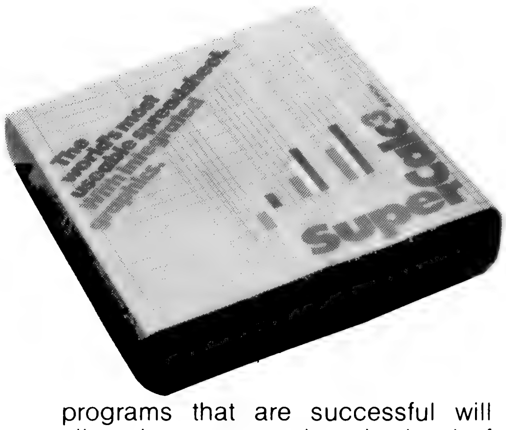
And they need to be fast, very fast. If the user is going to be kept hanging around, the user will not buy the program. Which is why Lotus 1-2-3 was a great success and MBA, on which it was possibly based, was not. Lotus 1-2-3 is a relatively fast program. MBA was not, although it has since been totally rewritten in machine language. The new version, which we have not tested, apparently performs like a Formula I racing car.

Simplicity still has a place

There is no doubt VisiCalc and SuperCalc III will remain perfectly viable for some years to come, especially for the simpler forms of programming.

Indeed SuperCalc III has now expanded so you can use a graphing program which is already integrated. And MicroSoft's Multi-Plan has a lot of mileage left.

Nevertheless the future of the spreadsheet is going to be in programs which contain at the minimum a functional database, a word processing program with some relatively sophisticated features, a total graphing system and a spreadsheet program which is completely flexible and contains selectable mathematical functions. And the


The world's most useable spreadsheet with integrated graphics.
SuperCalc3

programs that are successful will allow the user to select the level of sophistication needed in each area.

If the spreadsheet needs to be continually accessing a large relational database then it will have the facility to graft dBase II or III on with no problems and with a complete compatibility of instructions and commands.

Memory problems

The big problem is always going to be that such a program will take a very large amount of space on the computer.

Until recently, when we were all stuck with the theoretical limit of 64K on an eight-bit computer, this problem was almost unsurmountable.

It is almost impossible to cram such a program into 64K of memory, although brave attempts have been made. The only apparent way to make such a program operate was frequent access to the disk to shift pieces of the program in and out of the memory. This is far from ideal. It wears out the disk. It takes up time. And it allows errors to creep in.

Big memories

For the sort of advanced spreadsheet total concept program we have in mind, the computer will need at least 256K of Random Access Memory on board and very possibly something in excess of half a megabyte. This is already possible with the IIe using extra memory boards. It is, of course, axiomatic that the Macintosh and the Lisa were almost designed for such a configuration.

The traditional spreadsheet will be with us for some while but the way of the future is now quite clear – totally integrated programs that can be modified to suit precisely the user's specific needs.

Watch out for them Real Soon Now. □

Snapshot gives you another picture, Wildcard gives you an extra ace

by Duncan McCann

Before the review, first a rundown on morality; software piracy is an obvious and deliberate criminal act and is now punishable by law. Notwithstanding this fact, pirates rarely accept or understand the risk they run. Piracy proliferates in both hardware and software. I read recently that Atari managed to stop the sales of a ROM-copying device . . . primarily because the copier had no secondary purpose.

But what happened to the convention that one is allowed to make back-ups for one's own personal use? Not that I believe the ROM copier was being used so carefully. What difference exists between backing up software from a floppy disk or from a ROM?

No one is quite certain of where the law stands. But there seems to be a consensus of opinion. It is fair enough to make a back-up copy of a program for your own use but not to distribute, sell, or give away programs in any way whatsoever.

This means, in our opinion, that Wildcard, Snapshot and ReplayIIF among others (all hardware copying devices) and software such as the infamous Locksmith, Nibbles-Away, Crazy Copier will never be outlawed. Because they have a real and legitimate purpose – to provide back-up copies for the original purchaser. (Anyone who believes they will be used exclusively for this purpose needs their head read, but it does provide a legal justification for the existence of these devices and programs).

Wildcard

Before we look at Wildcard and Snapshot, perhaps I should mention how I came across the idea of such a card.

I was in England and saw an advertisement for a company called "Dark Star" which made this remarkable card called "Snapshot". Its technology is very similar to Wildcard and, indeed, my opinion is the two may originate from the same fertile brain.

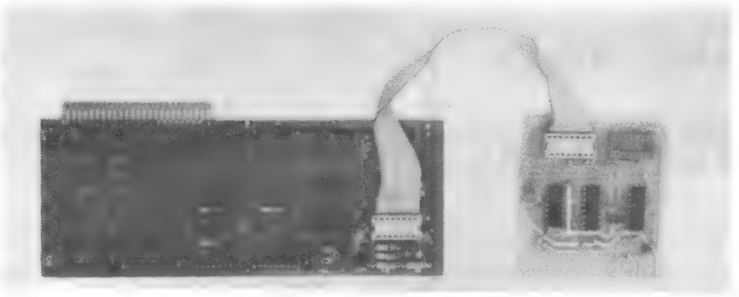
The advertisement gave the impression of high technology and there was a telephone number where you could make enquiries. I telephoned the number of this company on the cutting edge of modern technology, and a lady's voice said, "Could you call back after lunch, luv, he's still in bed."

In fact I telephoned two days later and found that one half of Dark Star had gone to the USA to negotiate a deal (which may have been the genesis of Wildcard) whilst the other half – his girlfriend – was staying at home answering the telephone and, neat touch this, rubbing the numbers off chips so the board could not be copied. She told me her boyfriend designed the board and she put them together.

That was two years ago. No doubt the situation has improved since then.

What it does

Both Snapshot and Wildcard are electronic circuits. The way they are



The Wildcard – Legal! But moral?

set up is slightly different. With my version of Snapshot I have one 16K RAM card that has to fit into slot Ø and a small card that can fit in any other convenient vacant slot. It is intended for an Apple II+ as it gives an extra 16K to bring it up to 64K.

The Wildcard slots into any 64K Apple II or //e. They are both designed to make copies of software loaded in RAM. Once installed, the program to be copied is booted or loaded in the normal way . . . then with the push of a button (like the button and tube that extends from professional studio cameras on the Wildcard, but using the paddle controls on the Snapshot) the process begins.

I won't describe the whole method here, because it is adequately described in the manuals (much improved over the earlier one for Snapshot which was typed out by the aforementioned girlfriend). The idea is for the board to find the program in RAM and then to create a bootable disk containing the plundered program.

Normally easy

Making copies of programs that run in 48k or less is usually easy, unless you have a combination of graphics, text and one or more slots initialised at the time the copy is taken. You will then also have to make choices from the options, which may not be that easy for people unfamiliar with Apple hardware.

64K copies need two takes. First, while the card works from the upper 16K, it copies the program hiding in the lower 48k reaches, and then again the other way around.

Naturally, for the two halves to ever work synchronously again, the moment chosen to make both copies must be identical. That's usually easier than it sounds. Simply choose any convenient stop in the program. Neither card can copy anything over 64k.

Quirks

There may be more quirks than those I've encountered, but say for example you copy VisiCalc on a 64k system that shows 34k of available space; you'll end up with what appears to be a working copy of that software. However, if you try to load a model that would reduce the counter below 15k (ie to use the upper reaches of 64k machines), then the system hangs. For some reason, you need to clear an apparently already clear screen and to connect the additional 16k before you can use the full 34k indicated by VisiCalc.

One area that I have never clearly sorted out is why copies made using either of these cards tend to be less satisfactory than copies made with other, software-driven, methods.

Time and time again I have had a copy fall over after only being used three or four times. It very much depends on the program you are copying, but these cards are not universally succesful and do not always make satisfactory copies, no matter what the advertisements say.

The manual also says it is difficult to copy software that does not load itself completely into RAM (ie reads more files during the normal use of the program such as PILOT, and nearly every other commercial program).

Turnkey copies

In practice, if the files are stored on disks initialised with DOS 3.3 the copied program will probably still work. Wildcard and Snapshot create turnkey copies on a blank disk which they first initialise with a DOS based on 3.3 . . . and this is why they will usually read ordinary DOS files.

Another feature common to both will allow you to make BRUNable files that work under normal DOS, instead of booting the turnkey disk made by Wildcard with its multitude of binary segments and unusual DOS. Not so useful, I think, but interesting, is an option to recover Applesoft from the copied program. If successful, you'll get statistical information about where the program starts and ends, and therefore also the length and runtime flag.

Also included in the utilities are a screen viewer and dump facility, the ability to patch vectors and pointers to modify the RESET vector and to locate and save the RWTS and DOS areas of memory.

All in all, Snapshot and Wildcard both require some effort to make successful copies. This makes me think they are unsuitable for businessmen who are keen to back up commercial programs with minimal fuss. It is more likely that an Apple enthusiast would buy Wildcard or Snapshot. □

ProDOS – the new operating system

Last year Apple introduced ProDOS as a new operating system for the Apple II family which would complement but not supplant DOS 3.3 of blessed memory.

This new operating system represents a significant advance over previous operating systems. But it is important for all Apple users to realise the difference affects programmers far more than it does users. Users will see benefits in speed and flexibility but no, repeat no, systems or programs will be outdated by the advent of ProDOS.

The story from Apple headquarters in Cupertino – from whom all wisdom flows – is that "software developers have expressed great interest in a standard operating system for the Apple II that produces advanced capabilities and such applications as networking, business graphics, and large database management. ProDOS is our response to this need." American business jargon at its finest.

Greater sophistication

This appears to mean that more sophisticated programs can be written to take full advantage of mass storage, specifically hard disks, and multi-user systems will be available for the Apple II family of computers.

This seems logical.

Apple are more convinced than ever that a large part of their fortunes for the forseeable future are tied to the Apple II series. Continuing upgrades will be the order of the day.

There are two versions of ProDOS. The developer's version which is 1.Ø D and the end user's version, that's you and me, called version 1.Ø1.

To use ProDOS to its full advantage, programmers need the ProDOS Assembler Toolkit plus a machine code debugger called Bugbyter.

Programs using ProDOS can be set up in one of two ways.

Either the programmer can expect the user to load ProDOS into the system – which seems fairly silly – or the programmer can build it in to the program, invisible to the user, which seems far more sensible.

Using utilities

Probably the only time the average user will ever load ProDOS with malice aforethought is when programming and using certain utility programs. Otherwise ProDOS will be built in to most application programs and automatically loaded on booting.

One of the stated reasons for the creation of ProDOS was to make it possible for the Apple II to support mass storage beyond the 14Ø kilobytes provided by the Apple II disk drive.

It was the users, you and me, who created the pressure for the Apple II to show it could support data communication and networking as well. Perhaps we had more faith in the Apple II then Apple did.

DOS 3.3 was never designed to handle these facilities because it was never imagined it would be used in this way.

ProDOS is completely new.

It is not a modified DOS 3.3.

Widely usable

ProDOS will run on any Apple II with at least 64K memory (that means any Apple IIe or Apple II plus a language card.) Note that ProDOS goes into the RAM card or its equivalent so that there is no room for Integer – not a great loss. When Applesoft Basic is required the Applesoft ROM must be installed on the motherboard as in the Apple IIe.

Most programmers we have talked to consider that ProDOS is a vast improvement on DOS 3.3 from their point of view. At the moment the choice of languages available is limited to Basic and Assembler

The ProDos User's Kit.

which to some may appear to be worse than no choice at all.

Faster running

Users will find that programs which use ProDOS run faster because of the more efficient use of memory and infinitely more efficient way of managing files.

ProDOS maintains as much compatibility as possible with DOS 3.3 when it is being used with Applesoft Basic.

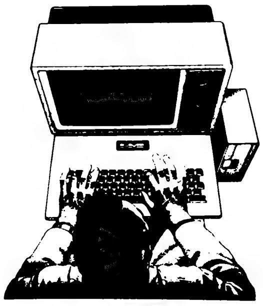
Interestingly, ProDOS was originally seen as a logical step up to the Apple III and its SOS. As the Apple III has now gone to that great computer graveyard in the sky, any step upwards to SOS (such an aptly named operating system) would not, in itself, be a good thing. The III has gone but ProDOS remains.

Main improvements

The main improvements of ProDOS are its hierarchical filing system, its support of mass-storage devices and its support of a defined interrupt protocol. (A defined interrupt protocol is an almost essential feature of any efficient networking system. During World War II there was a British radio show called ITMA which had a catch phrase "After you, Claude. No, after you, Cecil." Which is a early form of defined interrupt protocol. It lays down the rules as to which machine has priority at any given point in time in a network.)

Defined interrupt protocol is ▷

Apple Basic programmers' aid



If you have ever written a program directly into an Apple, you will realise exactly how difficult it is. If you are going to do any serious programming, you need a program line editor. The one we are using at the moment is the Global Program Line Editor (GPLE) which works extremely well.

When you boot the disk, the only way to tell whether the program has been loaded is a selection menu on the screen. If you have an Apple II+ with a 16K card in slot Ø, the program is loaded automatically onto that card.

Otherwise the program locates itself just below DOS. The program has several features.

First of all, it allows easy modification of program lines.

Secondly, it allows a series of characters to be defined and then executed with only two key strokes.

Thirdly, there is a listing control which gives you the capability to stop, start, end a listing to the screen.

Then it allows you to type in lower case letters and at the same time it gives you totally programmable cursor control.

The latest listed edition allows you:

* Global editing, including search, edit and replace routines.
* The ability to use 8Ø-column display on several popular add-on boards including Vision-8Ø.
* A type ahead buffer which stores characters when you are writing a program.
* A utility program which allows the automatic numbering of Apple softline.

With this program you can do some elegant pieces of editing, including auto-packing which is extremely useful for removing unnecessary spaces if you have too long a line.

You can also convert your program into, as it were, a book and read through it at a rate of twenty lines at a time. This is one of the useful features of the program.

The great virtue of Global Program Line Editor is that you actually do not notice it is there at all. You only use it when you need some of its capabilities. There is good documentation provided and extremely easy directions for use of the program.

Global Program Line Editor is produced by Synergistic Software, and it shows the intelligence of the manufacturers that they have left the program unlocked so you can copy it, if necessary.

If you are doing any serious programming in BASIC then you absolutely must have a line editor. Of those I have tested, the Global Line Editor is currently my favourite. However, John Lehane has just shown me this new program . . . □

---

almost essential if you are going to use hard disks. It is absolutely essential if you are going to network.

Because ProDOS uses the hierarchical filing structure you can keep files in sub-directories and organise your stored data far more efficiently than in a single level directory structure. This filing structure is not a million miles away from that used in Unix.

It means that you have a directory, then a sub-directory, then a sub-directory and if necessary yet another sub-directory.

Large files

ProDOS allows the Apple II to handle large files up to 16 megabytes in size, which, when you consider we are talking about an 8 bit machine, sounds faintly ridiculous.

The hierarchical filing structure gives the hard disk user an organised system of getting and accessing an immense number of files.

Let us take an example.

Supposing you had a five-megabyte hard disk attached to your Apple II. DOS 3.3 would not allow you to store information into a file larger than 14ØK.

ProDOS frees you from this restriction and, with its hierarchical filing structure, allows you not only to store it more efficiently but access it more efficiently as well.

There are some DOS 3.3 commands which are not supported in ProDOS but there are many new ones which enhance ProDOS immensely.

End result

The end result will see an immense amount of new application software taking advantage of the larger data storage and interrupt capabilities.

We are going to see local area networking, data communication, animation, word processing, spreadsheets and other applications set up, so they can use to the full the facilities of hard disk drives.

It is most important that users realise the difference between ProDOS and the standard DOS merely makes the programmer's life easier and, as a result, we will see faster, more flexible programs. But the vast ocean of programs that have been written in DOS 3.3 will continue to function. And new user programs will have ProDOS built-in. All of these programs will be self-booting.

And when new programs come along – like the Lotus 1-2-3 clone for Apple which we are promised in the near future – it will be very easy to convert DOS 3.3 text files to ProDOS text files, and, indeed, vice versa. □

In search of a great game

We often find ourselves very bored when reviewing Apple games. They all seem to be variations on either adventures or shoot-'em-up space wars.

Now we have found a program which is a genuine educational adventure game AND interesting and exciting.

Called "In Search of the Most Amazing Thing", it was created by Tom Snyder Productions for Spinnaker Software, distributed in Australia by Imagineering.

This is an educational adventure game set in a world called Porquatz. Half of the world is fairly dull and the other half is covered by a mere mist which is filled with deadly crabs, fascinating tribes and erratic characters.

The game lets you explore this world on a quest for a hidden object and you are given clues as you go along. What raises this game above the level of almost all others are the graphics, which are delightful and exquisitely done.

You are in a non-violent search for an object called the Most Amazing Thing, lost by Uncle Smoke Bailey many years ago. Uncle Smoke wants you to return the booty to his home in Metallica, a city in the Dark Zone Mire.

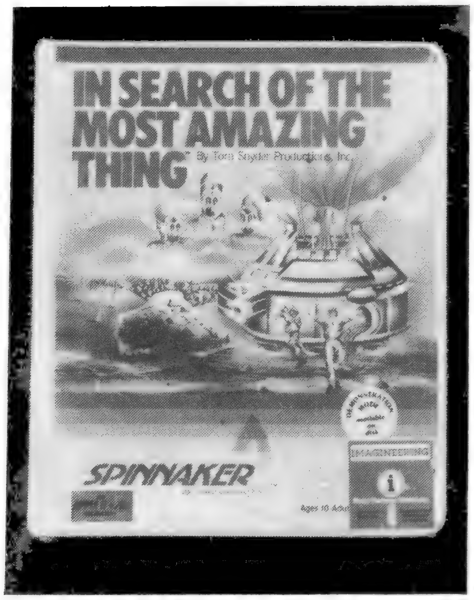
To help you he provides a Beeliner which is partly a hot air balloon and partly a dune buggy. You have to fly through storms, drive over land covered with bogs, deal with aliens and even compose music before you find the Most Amazing Thing.

Uncle Smoke is always ready to give you plenty of advice.

The start

You start off standing next to a trapdoor and when you enter it you find you are in an elevator of the underground city, Metallica.

You can either go to Uncle Smoke's apartment or to the galactic store or to the great Metallica auction.


IN SEARCH OF THE MOST AMAZING THING
By Tom Snyder Productions, Inc.
SPINNAKER
IMAGINEERING

After that it gets really complicated.

However, all the time you are playing the game you are given hints, ideas and directions which will help you, with a modicum of intelligence, to recover the Most Amazing Thing and bring it back safely.

This is just not a normal game. This is software within software within software.

There is a map which locates every hut in the mire. The program tells you all about the people of each mire culture. There is a dictionary which translates important words and phrases used by all the 25 cultures in the Dark Zone Mire. There is a program called Musix, that shows you how to create the songs that are used for trading with the mire cultures.

I am assured that most adults who know about playing computer games can complete this adventure in 1Ø to 12 hours. For us it just went on and on and on. □

Repeat sessions

Once you have played the game you can play it again because the creator of this immensely enjoyable fantasy game, Tom Snyder, changes the location of the Most Amazing Thing in every game that is played.

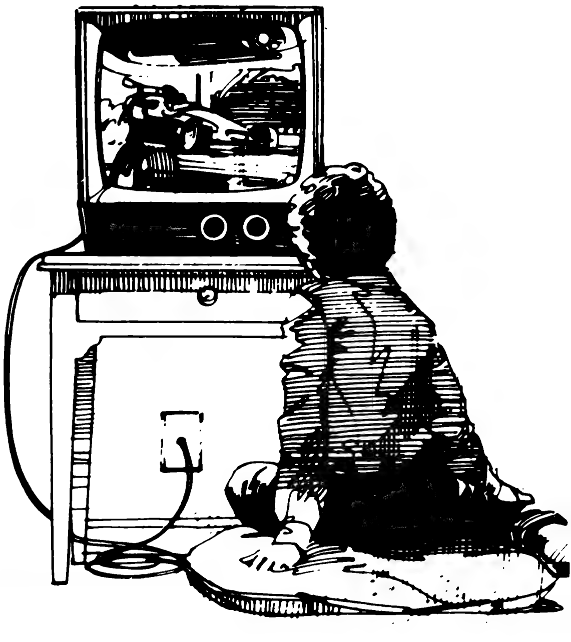
It is most reassuring and delightful to see that violence is not necessary to make an adventure game interesting.

This is an educational game, a game that requires the keeping of careful records and organisation of the records for quick reference.

It involves the use of basic mathematics, map-making and map-reading skills and an understanding of the differences in cultures and the need to make good decisions. Finally, In Search of the Most Amazing Thing also encourages reading by using a story book.

It would be difficult for us to recommend In Search of the Most Amazing Thing too highly. It is possibly the best adventure program ever written for computers.

Certainly it shows a direction in which computer games can now travel. □



Make your Apple go faster with Accelerator IIe

by Elizabeth Chamberlain

For those who use the Apple //e for number crunching, the Accelerator IIe card could – but not necessarily should – prove a godsend. And please note carefully we said that it applies if you use it for number crunching.

The advertisements in the United States boast that it "will make your Apple //e run three and half times faster" – indeed, it says it on the carton the board comes in – and the advertisements are indeed true as far as the programs I tested – alas, none of them involved serious number crunching when the speed increase should be far greater.

The board is easy to instal and within seconds you are away and racing.

That term is not used lightly as the first program I tried was a game program called Autobahn. If you don't know it, you have to drive through heavy traffic on a freeway where a lethal ambulance has right of way and where you encounter lots of other traffic, sudden narrowing of the road, tunnels, oil spillages etc.

It is not a game at which I excel, but even those friends with higher skills wiped out almost immediately.

To see the images zoom around at high speed is very funny and produced a lot of laughter. However, this is hardly a convincing demonstration of the utility of Accelerator IIe.

For that it took Sargon II. This is, in case you don't know, probably the best chess program available on Apple. It has seven levels and provides a good contest for beginners through to pretty good players. The problem is that above level 4 the response time is slow and one can wait 15 minutes or so for the computer to make a move.

The Accelerator IIe brings this down to a tolerable span and increases enjoyment dramatically.

Technicalities

The board is based on a fast (3.6MHz) 65Ø2 microprocessor with its own memory. This processor takes over all calculations, while the Apple's own 1MHz 65Ø2 controls only the video display.

Pre-booting

The 64K of memory includes a built in language card. For maximum efficiency, the program you run must be in the Accelerator's memory, so a pre-boot disk moves the language from ROM to the fast Language Card on the board.

However, with programs (such as VisiCalc) which run in the 48K of main RAM, the Accelerator automatically grabs everything from RAM, so pre-booting is unnecessary.

As the Accelerator is a direct memory access co-processor, it cannot be used at the same time as the Microsoft Z-8Ø Softcard, which does the same thing with a different type of co-processor. When you run CP/M software on the Z-8Ø Softcard, therefore, you need to turn off the Accelerator.

However, if you are using a Z-8Ø card such as Personal Computer Products 'Appli-card', which does not use direct memory access, this does not apply.

Impressive

The board is certainly an impressive looking beast. It is about the same size as an ordinary board, but crammed with chips – 43 of them plus the 65Ø2 microprocessor, the DIP switch block and a mysterious little gizmo simply labelled Saturn Systems.

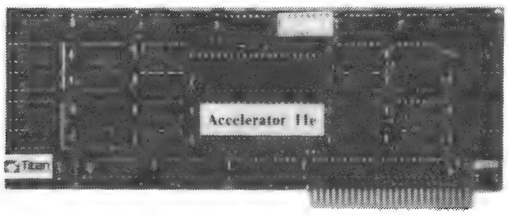
The fear with putting so powerful a board into one's Apple //e, especially when most of the slots are already occupied, is that the power supply may prove inadequate, or that the whole shebang may overheat.

The first proved no problem, and the cooling fan coped with the second.

I do not consider that games, even Sargon, are a fair test of the board's ability: one really needs to try it out on a number-crunching program. However, Accelerator II clearly does speed up processing dramatically and professional users especially should find it a good investment.

Editor's note: Far from me to quibble with a reviewer but the whole point regarding these go-faster boards is they are only relevant if you are seriously into number crunching. Most time lag in programs comes from disk access – not from deficiencies in the speed at which the CPU works. To take the greatest advantage from such a go faster board you need to load the program into the board – which is standard in most cases – and you need to load the data you are manipulating into a RAM pseudo disk card. This prevents any access to disk being required while you are working with the program and makes the whole setup operate very quickly. □

Distributed by Compumusic, 103 St. Johns Rd, Glebe 2307. (02) 692-9293.



The Accelerator IIe "turbo-charger".

Microsoft Basic for the Macintosh

The Macintosh – now with MBasic.

MicroSoft Basic was taken a step forward with the Apple Macintosh to reflect the increased power and the special features of the machine. At the same time it managed to maintain compatibility with earlier versions of MS-Basic.

The new Macintosh version is different in two ways.

First, it is easier to use. Because it runs on the Macintosh, it uses the pull down menu and the window and mouse-oriented user interface as do all Macintosh applications.

Second, it is much more powerful. The language supports strings up to 32,767 characters long. All numeric versions are assumed to be double-precision. The decimal math packages posts to 14 decimal points. All of this reflects the potent and inherent power of the 68ØØØ, the 32-bit chip microprocessor at the heart of the Macintosh.

When writing programs in MBasic you can make use of the Macintosh's windowing facilities. For example you can have one window showing one section of your program and another window showing a sub-routine.

You can scroll backwards and forwards in one window while the other remains stationary.

Long file names

Not only can you have massively long strings to the point of excess, you can also have file names 255 characters long. This is as long as a string in normal MS-Basic. Why you should want such a feature beggars the imagination, but effectively there is no limit to the length of your file name. Unless the only language you know is spoken by the Abenaki Indians on the West Coast of Canada or you wish to program in Welsh.

Arrays can have up to 255 dimensions and both numeric and string variable names can be up to 4Ø characters long.

Although you cannot use a reserved word as a variable name it is possible to use a reserved word embedded in a variable name.

You can access all the standard goodies on the Macintosh including the clip-board. Which means you can use the Mac's cut and paste abilities to transfer output from your Basic program to another Macintosh application.

Graphics supported

This new MS-Basic has a host of statements to support Macintosh graphics. An option allows you to define and draw a hollow or filled box with one command. The circle statement lets you draw hollow and filled circles or ellipses. The get and put statements also have graphics applications.

When you have finished messing around with your graphics you can use the L COPY statement. This will send an exact image of your screen to your image printer.

The standard MS-Basic statements are very similar to their counterparts from earlier versions, although now there are several more reserved words to take full advantage of the Macintosh's capabilities.

As a language MS-Basic is remarkably easy to use and gives limited support to the Macintosh Toolbox, which is in the ROM and contains useful sub-routines – 41 in all.

Not total usage

It is fair to say that MS-Basic does not use all the power and the advantages of the Macintosh.

It does not let you create your own pull-down menus, nor does it support all of the Macintosh Toolbox routines.

For that you are going to have to wait for Apple's own MacBasic which becomes available in the next few months.

However MS-Basic is a logical step forward and it exploits fully the Macintosh's 68ØØØ central processing unit.

Best of all it retains compatibility with previous generations of MicroSoft Basic, which means thousands of existing MicroSoft Basic programs can be run virtually unchanged on the Macintosh. And that is a major plus indeed.

Up in the dumps

by F. Walraven

Here is a simple graphics dump program that can be called from BASIC to produce a C.Itoh 851Ø-printed copy of any picture on your hi-res screen. It can also be adapted to other printers.

One often wants a printed form of a high-resolution graphics image produced on the Apple II. While this can usually be achieved using a commercially available graphics utility, not all of these can be called from a program without user input.

This simple graphics dump program written in 65Ø2 machine code is intended for an Apple II microcomputer together with a C.Itoh 851Ø printer. The program can be called from a BASIC program and produces a horizontally oriented copy of the image on hi-res screen 2.

The program has purposely been kept simple and no options such as scale changes, inverse printouts, etc have been incorporated. With some changes, it can be adapted to most other printers which allow addressing of individual print head pins.

Program philosophy

An Apple II hi-res screen consists of an array of 28Ø dots horizontally by 192 dots vertically. Each dot is represented in the Apple's memory as a single bit and shows up white when the bit has the value 1 (on) or black when its value is Ø (off). The bits (dots) are arranged in the Apple's memory as 7 bits to a byte; the value of the remaining bit (the high-order bit) determines the colour of the other 7 bits (dots). Thus the 28Ø bits representing a single horizontal line on the high resolution screen are represented in memory by 4Ø successive bytes (4Øx7=28Ø).

To produce an image of the high resolution screen on the printer it is simply necessary to print a dot wherever the bit corresponding to the screen dot has the value 1.

While this could be done line by line, 192 times, to cover the entire screen, this would be slow, cumbersome and lead to needless wear and tear on the printer.

Dot addressing

The C.Itoh printer allows addressing of 8 of the 9 pins of the print head when in the bit image graphics mode; thus 8 horizontal lines of the screen image can be printed at a time and this is done here.

In the bit image graphics mode the C.Itoh printer expects a command specifying bit image graphics followed by a number of bytes equal to the width of the image to be printed. Each byte contains 8 bits representing the individual dots of 8 successive screen lines. The rightmost bit (bit No Ø) addressed the topmost pin of the print head, but 1 the second pin from the top, etc. In this program, the bits of 8 successive screen lines at a time are converted into a sequence of 28Ø bytes which are then sent to the printer together with the required bit image graphics command.

This process is started at the top of the high resolution graphics screen and repeated 24 times (24x8=192) to cover the entire screen. In doing so the program changes the original position of the bits and thereby destroys the hi-res image. Therefore, before transfer takes place, the contents of the screen image memory section is copied to another section of memory and the original image is not affected.

Operation

A disassembly of the graph dump program is given in Listing 1 and the corresponding machine code is shown in Listing 2.

The graph dump program consists of a number of sections. The main program starts at location $1E9Ø (the $ indicates hexadecimal notation) and takes care of the rearrangement of the screen image bits into a form suitable for transmission to the printer. It calls a number of subroutines used for setting up the printer (located at $1E5Ø), resetting the printer for normal use ($1E72) and for sending a line to the printer ($1F6Ø). An output buffer used to store the 28Ø bytes to be sent to the printer is located at $1DØØ to $1E1D.

The six-byte bit graphics command needed for the C.Itoh 851Ø is located at $1E1E to $1E23. The output buffer is written to and read from back to front (this simplifies programming) so that each output line is sent to the printer as "ESC S Ø 28Ø" followed by 28Ø bytes from the buffer.

Zero page memory locations between $ØØ and $1Ø are used and the program therefore starts by saving the contents of these locations on the stack and by also saving the CPU register contents (code at locations $1E9Ø to $1E9B). The code at $1E9F to $1ECB copies hi-res screen 1 onto hi-res screen 2 from where it is converted to printer format.

Eight byte blocks

The program operates on blocks of 8 bytes at a time, one byte from each of the 8 screen lines being transferred to the output buffer. Starting with the block containing the leftmost bytes of the topmost 8 lines of the screen, the 7 low-order bits of these bytes (that is bits Ø to 6) are transferred by shifting via the carry bit into the last 7 bytes of the output buffer (code from $1EF2 to $1F2Ø). Zero page location $Ø3 and the CPU's X-register are used as counters for the 7 bits and 8 bytes respectively. They are initialised by the code at $1EEC to $1EF1.

Note that the addresses operated on by the shift instructions ($1EF3, $1EF4, $1EFend $1EF7) are set up and maintained by the program itself. Locations $1 EF3 and $1 EF4 contain the address of the screen byte and $1 EF6 and $1 EF7 the address of the output buffer byte being operated on. For the first block of each set of 8 lines block the address of the topmost byte is obtained from the table starting at $1 E2Ø. This is done by the code at $1 EFE to $1 EEB. A table is required here because in the Apple II successive screen lines are not stored sequentially in memory.

Anti-purist

Note: the procedure used here is an example of self-modifying code, something frowned upon by purists. In this instance, however, it is very efficient and does not lead to undue complications. Its use is therefore quite justified.

Zero page location $Ø5 is used as a block counter and is initialised at $1 EDA and maintained by the code at S1 F32. After the required bits of the first block are transferred, the pointer to the next block is incremented and the operation is repeated. Once 4Ø blocks have been transferred, the output buffer is full and its contents are sent to the printer.

The process is then repeated for the next set of 8 screen lines. A counter for the screen line sets (zero page location $Ø4) is initialised at $1 ECC and incremented by the code at $1 F3D to $1 F44. Once the entire screen has been printed, the program switches output back to the screen, restores the zero page locations and CPU registers, displays hi-res screen 1 and returns control to the program that called it.

Program use

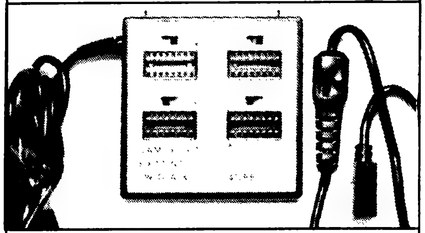
As noted in the introduction, the graph dump program can be used from within a BASIC program. After entering the machine code shown in listing 2 at location $1 E18 and $1 F93, the program should be saved using the command "BSAVE GRAPH DUMP,A$1 DØØ, L$2FF". Any program that is to use GRAPH DUMP should first load it into memory with the command "BLOAD GRAPH DUMP". The program can then be called from BASIC with the command "CALL 7824".

Note: this location of the graph dump program within the memory uses up space below hi-res page 1 which could clash with longish programs. The author has found it more convenient, however, to place long programs that use high resolution graphics above hi-res page 2. In this case placement of GRAPH DUMP below hi-res page 1 represents a more economical use of memory space.

Modification for other printers

The graph dump program should easily adapt for use with printers other than the C.Itoh 851Ø, provided the print head pins can be individually addressed. Changes would probably be required only in the code sections relating to the setting up and resetting of the printer. The bit image graphics command preceding the output line would probably have to be changed and the order in which the bits are transferred to the output bytes might possibly require changing. □

APPLE GAME PORT EXTENDER



Features

- ★ Four Switchable Gameports
- ★ Diode Isolation
- ★ Long Cord with Strain Relief
- ★ Plugs into Apple Gameport

ONLY $20 (+ $2 postage)

W. D. AUTOMATION & CONTROL
4 Emily Ave., Emu Plains 2750
After Hours (047) 35 5023

AUTO REPEAT

Just Clips on Inside

★ Apple II Plus ★ Europlus

ONLY $7 (+ $1 postage)

Send Stamped S.A.E. for Brochure

Program listing 1. Disassembly of the program Graph Dump.

```
1E50-   A0 F2       LDY   #$F2
1E52-   B9 6D 1D    LDA   $1D6D,Y
1E55-   20 ED FD    JSR   $FDED
1E58-   C8          INY
1E59-   D0 F7       BNE   $1E52
1E5B-   20 8E FD    JSR   $FD8E
1E5E-   60          RTS


1E72-   A0 FB       LDY   #$FB
1E74-   B9 86 1D    LDA   $1D86,Y
1E77-   20 ED FD    JSR   $FDED
1E7A-   C8          INY
1E7B-   D0 F7       BNE   $1E74
1E7D-   20 8E FD    JSR   $FD8E
1E80-   60          RTS

*
1E90-   20 4A FF    JSR   $FF4A
1E93-   78          SEI
1E94-   A2 0A       LDX   #$0A

1E96-   B5 00       LDA   $00,X
1E98-   48          PHA
1E99-   CA          DEX
1E9A-   10 FA       BPL   $1E96
1E9C-   20 50 1E    JSR   $1E50
1E9F-   A9 00       LDA   #$00
1EA1-   85 06       STA   $06
1EA3-   85 08       STA   $08
1EA5-   AA          TAX
1EA6-   A9 20       LDA   #$20
1EA8-   85 07       STA   $07
1EAA-   A9 40       LDA   #$40
1EAC-   85 09       STA   $09
1EAE-   A0 00       LDY   #$00
1EB0-   B1 06       LDA   ($06),Y
1EB2-   91 08       STA   ($08),Y
1EB4-   C8          INY
1EB5-   D0 F9       BNE   $1EB0
1EB7-   E6 07       INC   $07
1EB9-   E6 09       INC   $09
1EBB-   E8          INX

1EBC-   E0 20       CPX   #$20
1EBE-   D0 EE       BNE   $1EAE
1EC0-   2C 50 C0    BIT   $C050
1EC3-   2C 52 C0    BIT   $C052
1EC6-   2C 55 C0    BIT   $C055
1EC9-   2C 57 C0    BIT   $C057
1ECC-   A9 00       LDA   #$00
1ECE-   85 04       STA   $04
1ED0-   A9 1E       LDA   #$1E
1ED2-   8D F7 1E    STA   $1EF7
1ED5-   A9 17       LDA   #$17
1ED7-   8D F6 1E    STA   $1EF6
1EDA-   A9 00       LDA   #$00
1EDC-   85 05       STA   $05
1EDE-   A4 04       LDY   $04
1EE0-   B9 20 1E    LDA   $1E20,Y
1EE3-   8D F4 1E    STA   $1EF4
1EE6-   B9 38 1E    LDA   $1E38,Y
1EE9-   8D F3 1E    STA   $1EF3
```

listing continues

Repairing your Apple

When the Apple was first designed it was a marketing sensation for several reasons.

Firstly, it was the first totally self-contained personal computer within the strict meaning of that name.

Secondly, it was a brilliant single-board design – perhaps the best single piece of electronics design of its decade.

Thirdly, as a computer it was singularly easy to adapt and modify.

It did not occur to the original purchasers that because of its ease of adaptation and modification the Apple was also an extremely easy machine to repair and maintain. But they soon found out.

Repairing your Apple yourself, once it has gone out of warranty, can be a comfort and joy and can also save a bundle of money.

With the cost of repairing computers soaring by the day ($50 an

Program listing 2 continued

```
1EEC-   A9 07       LDA  #$07        1F25-   69 01       ADC  #$01        1F60-   A9 1E       LDA  #$1E
1EEE-   85 03       STA  $03         1F27-   8D F3 1E    STA  $1EF3       1F62-   8D 6C 1F    STA  $1F6C
1EF0-   A2 08       LDX  #$08        1F2A-   AD F4 1E    LDA  $1EF4       1F65-   A0 1E       LDY  #$1E
1EF2-   4E F8 43    LSR  $43F8       1F2D-   69 00       ADC  #$00        1F67-   A2 02       LDX  #$02
1EF5-   6E FF 1C    ROR  $1CFF       1F2F-   8D F4 1E    STA  $1EF4       1F69-   88          DEY
1EF8-   18          CLC              1F32-   E6 05       INC  $05         1F6A-   B9 00 1C    LDA  $1C00,Y
1EF9-   AD F4 1E    LDA  $1EF4       1F34-   A5 05       LDA  $05         1F6D-   48          PHA
1EFC-   69 04       ADC  #$04        1F36-   C9 28       CMP  #$28        1F6E-   8D 90 C0    STA  $C090
1EFE-   8D F4 1E    STA  $1EF4       1F38-   D0 B2       BNE  $1EEC       1F71-   A9 34       LDA  #$34
1F01-   CA          DEX              1F3A-   20 60 1F    JSR  $1F60       1F73-   8D 91 C0    STA  $C091
1F02-   D0 EE       BNE  $1EF2       1F3D-   E6 04       INC  $04         1F76-   49 08       EOR  #$08
1F04-   38          SEC              1F3F-   A5 04       LDA  $04         1F78-   8D 91 C0    STA  $C091
1F05-   AD F6 1E    LDA  $1EF6       1F41-   C9 18       CMP  #$18        1F7B-   AD 91 C0    LDA  $C091
1F08-   E9 01       SBC  #$01        1F43-   D0 8B       BNE  $1ED0       1F7E-   29 80       AND  #$80
1F0A-   8D F6 1E    STA  $1EF6       1F45-   A2 00       LDX  #$00        1F80-   F0 F9       BEQ  $1F7B
1F0D-   AD F7 1E    LDA  $1EF7       1F47-   68          PLA              1F82-   AD 90 C0    LDA  $C090
1F10-   E9 00       SBC  #$00        1F48-   95 00       STA  $00,X       1F85-   68          PLA
1F12-   8D F7 1E    STA  $1EF7       1F4A-   E8          INX              1F86-   C0 00       CPY  #$00
1F15-   A4 04       LDY  $04         1F4B-   E0 0B       CPX  #$0B        1F88-   D0 DF       BNE  $1F69
1F17-   B9 20 1E    LDA  $1E20,Y     1F4D-   D0 F8       BNE  $1F47       1F8A-   CE 6C 1F    DEC  $1F6C
1F1A-   8D F4 1E    STA  $1EF4       1F4F-   58          CLI              1F8D-   CA          DEX
1F1D-   C6 03       DEC  $03         1F50-   20 3F FF    JSR  $FF3F       1F8E-   D0 D9       BNE  $1F69
1F1F-   D0 CF       BNE  $1EF0       1F53-   2C 54 C0    BIT  $C054       1F90-   20 8E FD    JSR  $FD8E
1F21-   18          CLC              1F56-   20 72 1E    JSR  $1E72       1F93-   60          RTS
1F22-   AD F3 1E    LDA  $1EF3       1F59-   60          RTS
```

Listing 2. Machine Code Dump of the program Graph Dump

```
1E18- B0 B8 B2 B0 D3 9B 00 00    1E98- 48 CA 10 FA 20 50 1E A9    1F18- 20 1E 8D F4 1E C6 03 D0
1E20- 40 40 41 41 42 42 43 43    1EA0- 00 85 06 85 08 AA A9 20    1F20- CF 18 AD F3 1E 69 01 8D
1E28- 40 40 41 41 42 42 43 43    1EA8- 85 07 A9 40 85 09 A0 00    1F28- F3 1E AD F4 1E 69 00 8D
1E30- 40 40 41 41 42 42 43 43    1EB0- B1 06 91 08 C8 D0 F9 E6    1F30- F4 1E E6 05 A5 05 C9 28
1E38- 00 80 00 80 00 80 00 80    1EB8- 07 E6 09 E8 E0 20 D0 EE    1F38- D0 B2 20 60 1F E6 04 A5
1E40- 28 A8 28 A8 28 A8 28 A8    1EC0- 2C 50 C0 2C 52 C0 2C 55    1F40- 04 C9 18 D0 8B A2 00 68
1E48- 50 D0 50 D0 50 D0 50 D0    1EC8- C0 2C 57 C0 A9 00 85 04    1F48- 95 00 E8 E0 0B D0 F8 58
1E50- A0 F2 B9 6D 1D 20 ED FD    1ED0- A9 1E 8D F7 1E A9 17 8D    1F50- 20 3F FF 2C 54 C0 20 72
1E58- C8 D0 F7 20 8E FD 60 9B    1ED8- F6 1E A9 00 85 05 A4 04    1F58- 1E 60 EA EA EA EA EA EA
1E60- CE 9B BE 9B D4 B1 B6 9B    1EE0- B9 20 1E 8D F4 1E B9 38    1F60- A9 1E 8D 6C 1F A0 1E A2
1E68- CC B0 B1 B5 8D EA EA EA    1EE8- 1E 8D F3 1E A9 07 85 03    1F68- 02 88 B9 00 1C 48 8D 90
1E70- EA EA A0 FB B9 86 1D 20    1EF0- A2 08 4E F8 43 6E FF 1C    1F70- C0 A9 34 8D 91 C0 49 08
1E78- ED FD C8 D0 F7 20 8E FD    1EF8- 18 AD F4 1E 69 04 8D F4    1F78- 8D 91 C0 AD 91 C0 29 80
1E80- 60 9B BC 9B C1 8D EA EA    1F00- 1E CA D0 EE 38 AD F6 1E    1F80- F0 F9 AD 90 C0 68 C0 00
1E88- EA EA EA EA EA EA EA EA    1F08- E9 01 8D F6 1E AD F7 1E    1F88- D0 DF CE 6C 1F CA D0 D9
1E90- 20 4A FF 78 A2 0A B5 00    1F10- E9 00 8D F7 1E A4 04 B9    1F90- 20 8E FD 60
```

Repairing your Apple

hour is not uncommon) obviously anything that you can do to keep your Apple in reasonable repair yourself is a great saving.

We are not trying to put repairmen out of business, nor are we going to suggest that you fiddle with your Apple unnecessarily.

But there are ways and means of making sure that your Apple works without you having to drag it down to the repair shop every time it blinks and hiccups.

This series of articles is designed to help you get at least to the stage where as a happy home handyperson you can solve many of the basic problems that arise.

Four parts

The Apple was designed in four major parts.

1. The case.
2. The power supply.
3. The mother board.
4. The keyboard.

Let's take them one by one.

The case

The chances of anything going wrong with the case is almost zero. In all our experience with Apples (and that is probably a wider experience than most people's) we have never seen an Apple computer body crack or break.

The only major problem that can arise is if you use too much force to insert or take out screws. Then they can jam permanently.

All you have to remember is to use the right sized screwdriver whenever you are working on the Apple and not to use too much force. Brute strength and ignorance will create problems. Also, it is not a bad idea to clean the case occasionally.

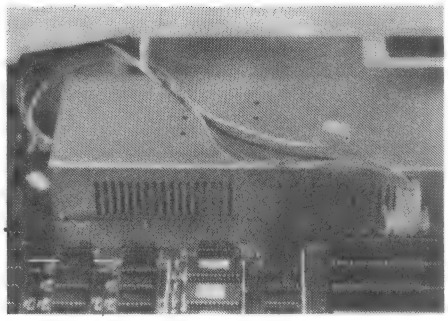
We use liquid Ajax which seems to work very well.

The power supply

If you are working with a genuine Apple power supply (as opposed to a fake or imitation Apple power supply) the only problem that you are probably ever going to have is with the switch.

The switch has a tendency to break down after incessant use.

How to fix it?



The silver or gold box inside your Apple, called the "power supply".

Open your Apple by lifting the lid at the rear of the case, pulling up and sliding it back and away from you. Put the lid in a safe place.

There on the left hand side is the power supply. It will either be silver or gold coloured. (If it's a different colour you may be dealing with a fake Apple in which case the situation changes slightly.)

The power input is at the back of the power supply and the switch is next to it. If your switch goes you do not need to take the whole of the power supply apart. In fact it is very difficult to do so, as Apple in their wisdom rivetted it together. That is not meant as a sarcastic remark. Once the power supply has converted the mains power to the low direct current the Apple uses, the chances of you getting a shock are minimal. But inside the power supply box there is a large amount of 240 volt alternating current floating around, which can be extremely injurious to your health and wellbeing.

Whenever you are working with an Apple, earth any static electricity which may be in your body by touching the top of the power supply box. You won't feel a thing, promise, but you will have been neutralised.

Then take the power plug out of the wall socket and out of the socket at the back of the Apple. If you think tha that makes us sound like old women, we don't mind. But it is truly much better to be safe than frizzled.

The switch is next to the power inlet and is a press-fit. You can ease it out with a flat-bladed screwdriver inserted under the edge.

Take it slowly, gently and easily. Rest assured that the switch will come out and that is replaceable with almost precisely the same switch available at your local Tandy's. Presumably your Apple dealer will sell you one but we have never checked.

Make careful note of the way in which the two connections lie on the switch, then unsolder them and resolder on the new switch.

Your new switch will now press back in the hole in the power supply and you are back in action.

Getting around the problem

There is another way of getting around this problem, and that is to use one of the hang-on fans which has a built-in power switch which will take the strain off the switch on your computer. Alternatively, you can use a substantial household switch inserted in the power supply cord and screwed to your desk or table near the computer, so that you can use that switch to power the computer on or off without using the switch at the back of the Apple.

The switch is one of the first parts of the Apple to go wrong and is, fortunately, one of the easiest of faults to rectify.

If the power is working you can nearly always tell because of the beeping sound that comes when you switch on, or the light that appears at the front of the computer.

Suppose you switch on, there is a beeping sound, the light on the keyboard glows and nothing else happens.

Your disk drive sits there and ignores you. No little red light comes up on the front, no whirry noises happen on the inside, nothing comes up on the screen.

The first step is to suspect any peripheral cards that you have in the machine.

As a first step, switch OFF – very important this – switch OFF and after you have switched off, take out every peripheral card in your machine. The lot. As you take them out note carefully with pencil and paper which slot they go into and the way in which the slots are numbered.

Eight slots

For absolute beginners, there are eight slots on the Apple motherboard which you can see as soon as you open the lid.

These slots are marked with maddening consistency from 0 on the left through to 7 on the right.

The Apple IIe, just to confuse matters a little, has slot 0 in the middle of the motherboard and its seven other slots in the standard position at the top of the motherboard.

These slots are one of the delights of the Apple – but also one of its weaknesses.

Into these slots go peripheral boards which allow your machine to perform in the various wondrous ways that it does. When you look down into these slots you will see a series of grippers which look as if they are plated with gold. And by damn they are. This is to preserve the best possible contact with the gold teeth of the peripheral boards when they are inserted.

Having removed all your peripheral boards and noted with great care where they came from and the way in which they were aligned, try switching on your Apple again.

If you get a satisfactory beep and the light comes on at the keyboard you know that the trouble lies with one of your peripheral cards.

Disk controller

Now, switch off again and have a careful look at your disk drive controller card that was in Slot 6 before you removed it.

At the bottom you will see a series of gold teeth which are there to be gripped once the board has been inserted. They should be shining bright. If they are not, give them a quick clean up with a pencil eraser. And do it away from the computer or you will be letting a lot of nasty dirty crumbs fall in your Apple.

At the same time make sure that the connector from Drive 1 is fitted correctly at the top of the disk controller board on to the set of prongs marked for Drive 1.

We found a machine three weeks ago in which the plug had been fitted on with one prong over at one end and one prong short at the other.

Make sure the plug is on properly and squarely pushed home.

We know you wouldn't be so daft as to make the mistake of putting the plug on backwards, but it does happen.

The flat multi strand wire is NOT crushed up against the disk controller card but is attached on the outward edge and is therefore not subject to any severe bending.

Trying again

With the computer OFF insert your disk drive card into Slot 6. (You may think we go on like a two bob watch about this switching on and switching off. But if you insert your disk controller card into your Apple when the power is switched on, you will stuff one of the chips permanently. In fact, you will blow a large and very visible hole in the side of one of the chips. As we have done this maybe half-a-dozen times ourselves we know what we are talking about.)

Recommended technique

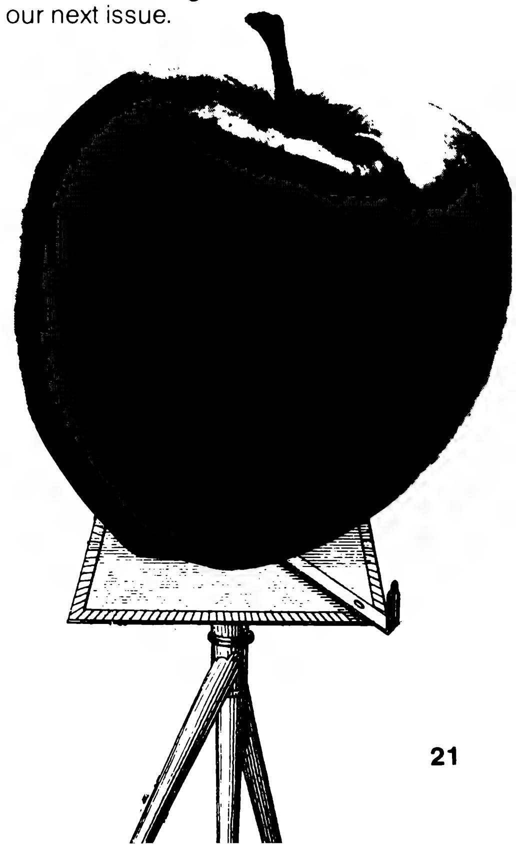
The recommended way of inserting the card so that it fits properly is to insert one end first and then rock the card down into the gripping teeth (they are spring loaded) until it is firmly and correctly in position.

In deference to our old eyes, we always use a torch to check that we have got it right before we go a step further.

Once you know the card is installed properly, and only then, switch on and the disk drive red light should come on and there should be whirry whirry noises and some form of life will appear on the monitor.

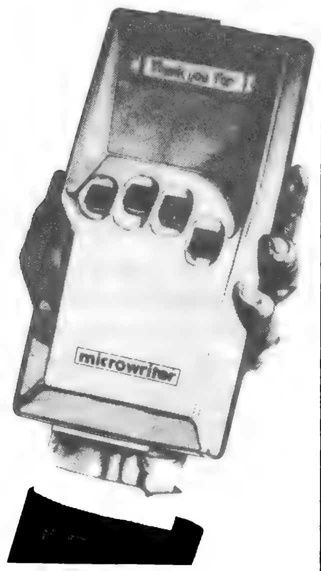
If this is the case one of your other peripheral cards is suspect. Either it hasn't been fitted properly or a chip has blown.

We'll talk about these and other matters involving the mother board in our next issue.



6 BUTTON TYPEWRITER AND WORDPROCESSOR

* simple to learn and easy to operate.
* no typing ability required – produce typed text the same day.
* world's most powerful hand held wordprocessor
* $790.00 including sales tax.



"... it's easier to learn than riding a bicycle ..."
The Australian

"The Microwriter is ... a replacement for every known method of putting thoughts onto paper"
Newsweek

RING TODAY AND GIVE UP WRITING!

PHM AUSTRALIA PTY. LTD.
(INCORPORATED IN NSW)

SYDNEY: SUITE 8, 5th FLOOR, 22 CENTRAL AVE,
P.O. BOX 696, MANLY NSW 2095
PHONE (02) 977 6600 TELEX 25015
BRISBANE: PHONE (07) 394 2388 TELEX 40280
PERTH: PHONE (09) 387 6396 TELEX 95995

microwriter

DEALER ENQUIRIES WELCOME

CAD on the Cheap..

Barry Kauler explores the fascinating potential of COMPUTER AIDED DRAFTING on his Macintosh.

Apple's Macintosh is one of the lowest-cost and fastest serious graphics machines around, which was one of the many intriguing features that drew me to it in the first place. I dreamed about the possibilities for draftspeople, education, artists, architects, engineers, etc, but at the time when I bought my Mac there wasn't much software: just MACPAINT, with vague promises of something else called MACDRAW sometime in the future.

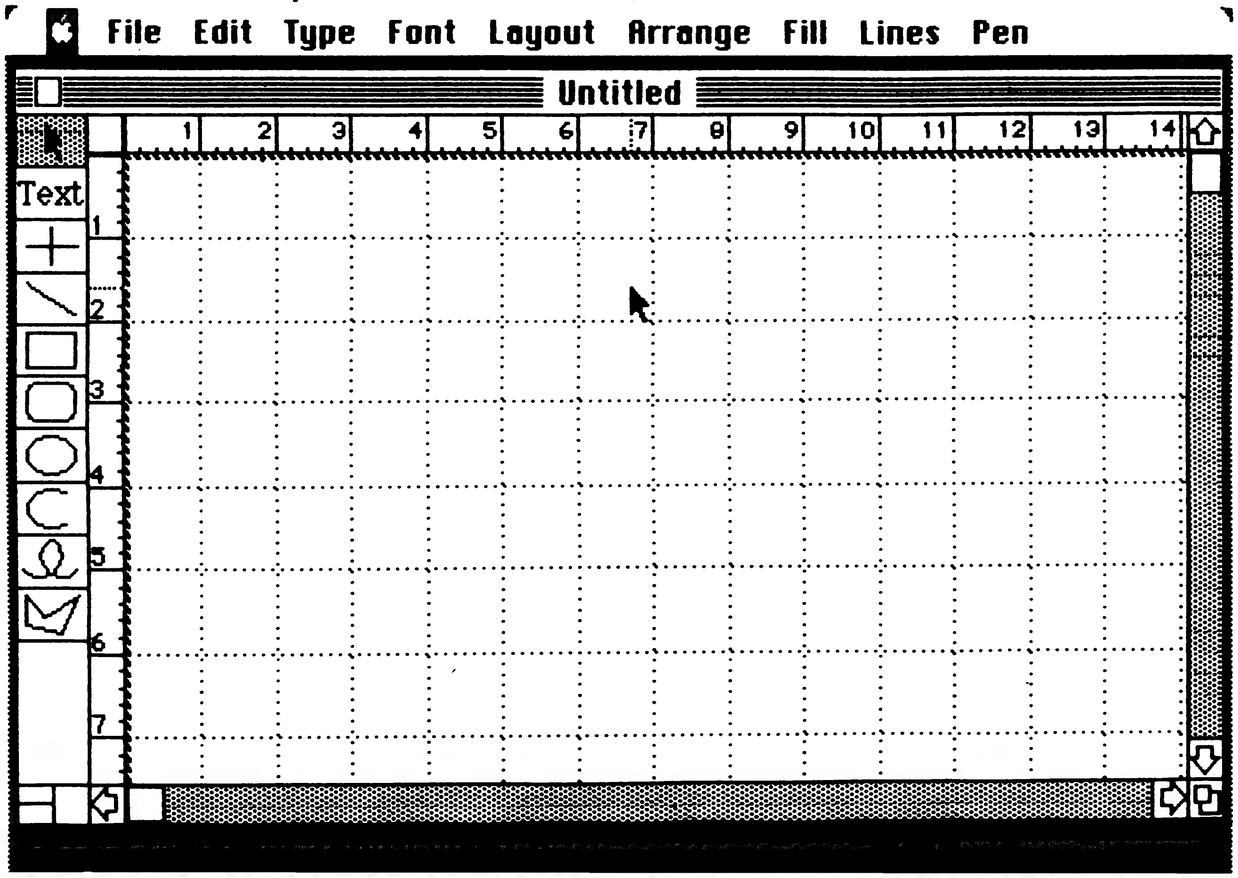
At the time of writing MACDRAW still hasn't been released, but I have been able to play on a pre-release version.....

and boy am I excited!

In this article I'm just fooling around, doing a few things with Macdraw, and hopefully you'll pick up the feeling I have for Mac's potential. Of course for a price around the $3000 mark (Australia) don't expect too much. For example the graphics resolution is a little bit coarse for serious professional work, at 512 by 347 pixels, but no worries, I found that particular limitation easy to live with. Some of the possibilities are so intriguing, such as the in-built Local Area Networking, which should allow a room-full of Macintoshes to be linked together and onto a common plotter. Perhaps also a Lisa computer with hard disk should be included in the network for a serious drafting system.

My own background is in education, so I tend to see things from that viewpoint: I can envisage the above-mentioned classroom-full of Mac's being used to teach people about CAD. Certainly it's the most cost-effective option I've ever encountered.

FIGURE ONE. What you see when first opening up Macdraw:



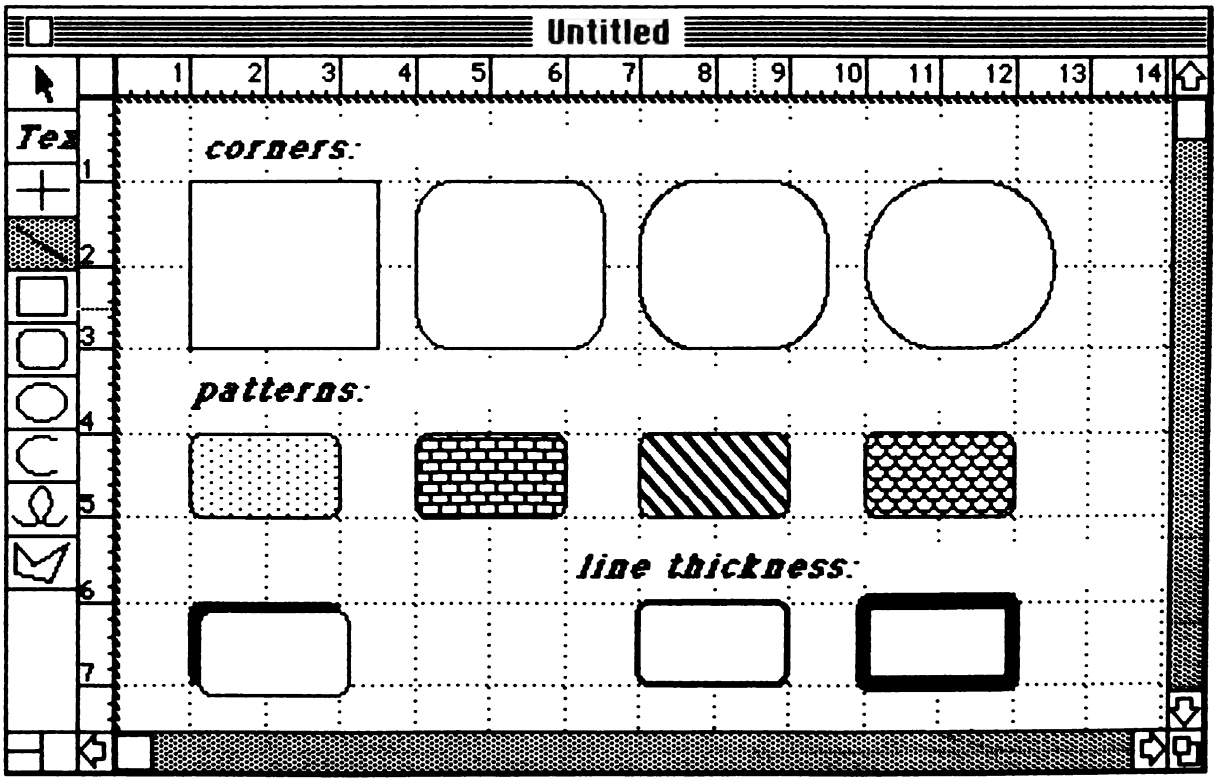
ABOUT MAC:

Firstly a brief description of Mac himself. his brain is a Motorola 68000. Memory is 64K of ROM operating system subroutines, and 128K of RAM, expandable to 512K. Mass storage is one 3.5 inch microfloppy drive holding 400K, with a socket for external microfloppy or hard-disk drives. Two 230Kbit/sec serial interfaces are for Local Area Networking, a printer or Modem. Other standard features include a mouse, battery-backed clock, and voice-synthesis.
As mentioned above the screen is 512 by 347 pixels, and is also monochrome only. Due to Mac's popularity in the U.S., a very wide range of software is becoming available.
Now for my personal comments about Mac. I'm currently running "Microcomputer Appreciation" short courses at the Collie Technical College in W.A., and I've been bringing my Mac in for the students to use. now this is fine as it gives them a chance to compare Mac with the other machines in the course, which are CP/M-based..... but there is now a problem, as Mac is so much **easier** and **fun** to use that there is a fight at every session over who gets to use Mac. That's one comment. Ease of use is the one main comment I want to make: it's my over-riding impression and also that's what impresses other people to whom I've shown Mac. But there are also some negative aspects, one of which is the limited memory. 128K RAM and 400K disk just isn't enough for this kind of machine. Solution: buy the RAM memory upgrade when it becomes available, and also improve disk capacity somehow. This latter requirement has a few options: buy an external floppy or hard disk, or wait until the double-sided microfloppy drives become available, and upgrade the Mac. The present standard Mac system is workable but a bit limiting and frustrating at times.

MACDRAW OVERVIEW:

Now for the heart of this article, which is to look into CAD on the Mac. Macpaint, which was supplied free with my Mac, along with Macwrite, is a fascinating piece of software, and is reviewed quite thoroughly in earlier articles in various journals. However Macdraw is still lurking in the shadows, at least at the time of writing. I'm not going to give a detailed description of all the ins and outs of Macdraw, as I find such reviews boring whenever I read any of them. Instead I'll just flit around and give you a taste of some possibilities. Firstly I must emphasize again that my version of Macdraw may not have all the features of what will finally be offered to the public.

FIGURE TWO. Playing with rectangles:



When Macdraw is first entered, a grid appears on the screen, with a menu bar along the top and another menu of drawing selections on the left-hand side of the screen. Figure One shows this. Notice also the rulers along the top and side: when the mouse pointer is moved anywhere in the drawing area, little **crosshairs** appear in the rulers showing the **exact X and Y coordinates** of the pointer. This enables precise positioning, but also Macdraw will compute the **horizontal and vertical components of any line, curved or rectangular shape**, in real-time, so drawings can be made to precise dimensions.
Mac's screen is so small, only 9 inches, but the paper size you're working on can be from 8 inches by 10 inches up to a huge **50 inches by 96 inches**, and the screen is like a window that can be scrolled around to look at any section of the drawing. There is a reduce-mode that allows viewing of the whole drawing on the screen, though of course detail is lost.
The rulers and grid can be imperial or metric units and the number of increments between numbered divisions can be selected. In this example shown in Figure One the divisions are in centimetres and there are 10 increments per division. An option can be selected to make any lines drawn, **automatically snap into the nearest increment**, which makes it easy to draw lines that meet each other exactly.
Any area of drawing can be **shrunk or expanded** as required and also **moved around** on the "paper", which is really handy.

Figure Two shows some varistions possible with the basic rectangle. The top row shows a rectangle with square edges, which was then duplicated three times and the copies moved to the side. So **duplication** of a selected shape is an available feature. Then I modified the radius of the corners, and the three different rectangles were produced as shown. So it'll also **round corners to a given radius**. Also there's a **curve** function in the left-hand menu, which can be used for such purposes, or anywhere that curves are required. For example a curved arch-way in an architectural drawing. Then from the **fill** menu on top I selected some patterns and put them into rectangles as shown. Then by drawing one rectangle on top of another, slightly offset, and filling the back one with black, I produced the shadow effect shown. A range of **line thicknesses** are available, and I drew a couple of rectangles with different line thicknesses.
All of these rectangles were drawn using the **rectangle** selection from the left-hand menu. All of these menu selections are so easy to use, and once something is drawn onto the screen, one can come back at any later stage and modify what one has drawn: the flexibility of this and other features is **so much easier to demonstrate than explain.**

FIGURE THREE. **Drawing that has been cut from Macdraw and put in the Clipboard:**

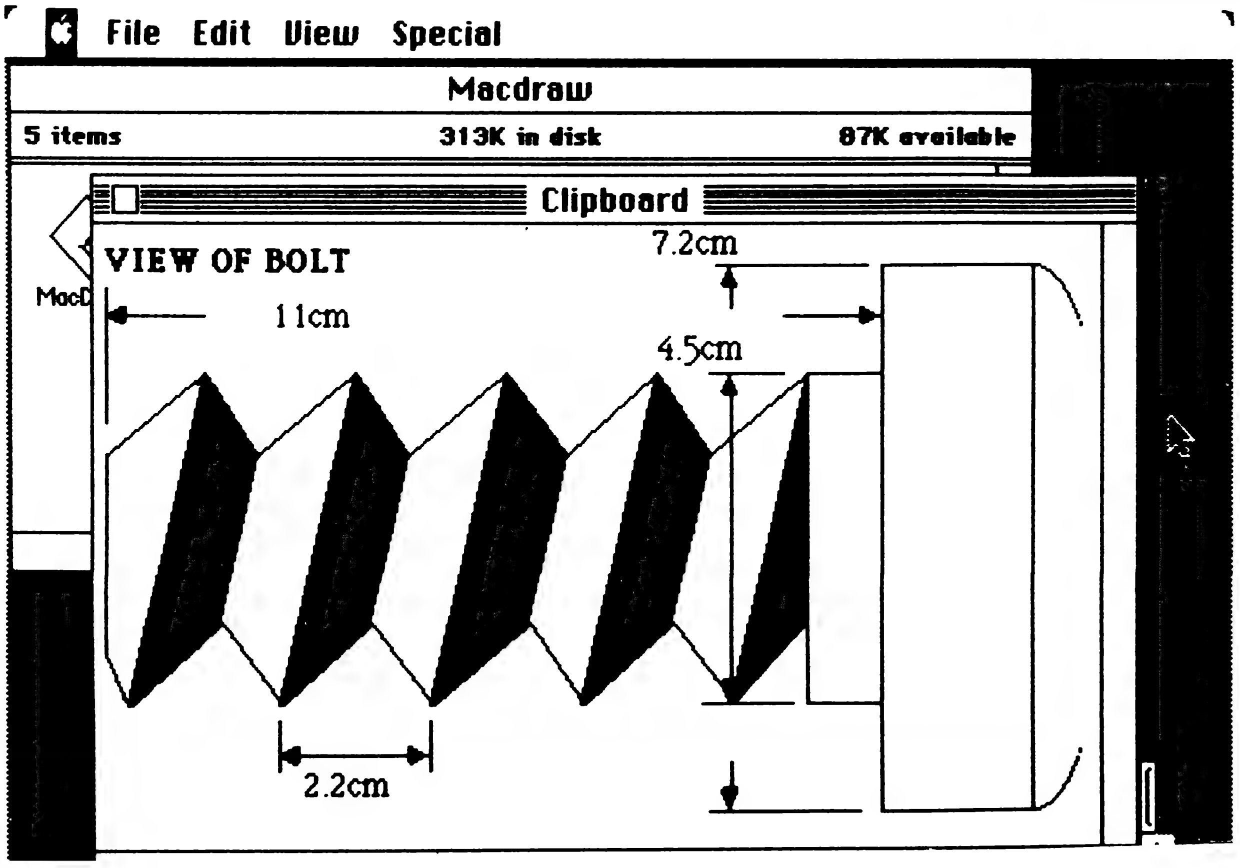


Figure Three is interesting. I drew a bolt using Macdraw, then I got the idea to include the picture in a report that I was going to write using Macwrite. One feature of Macintosh is that graphics and text storage is standardized, allowing easy transfer of information between different applications. The usual procedure is to select what is to be cut out of the Macdraw picture, select **cut** or **copy** from the **edit** menu at the top of the screen, which then places the picture into an intermediate storage area called the **clipboard**. After quiting Macdraw and opening up Macwrite, I can then select **paste** from the edit menu and put the picture into my report. It's quite easy. The clipboard itself can be viewed on the screen, and Figure Three shows the clipboard, behind which is the Macdraw-disk directory window.

APPLICATION EXAMPLE:

I'm an electronic engineer by qualification, so I had an interest in using Macdraw for such purposes, and I found to my pleasant surprise that it's a piece of cake. Figure Four shows an example. What I did was create a **menu of electronic symbols**, then these can be pulled out at will and dragged to the required position. Also they can be **rotated** as required. I placed the menu right at the top of the page. It's saved on disk so I can get it anytime I want to do an electronic drawing. However once any symbol has been dragged from the menu and used in the drawing, there's no need to go back to the main menu, as the symbol already placed into the circuit diagram can be duplicated and the copy dragged wherever required. In fact the menu could be pulled off disk to get going fast, then disposed of, that is, erased from the drawing.

What I found is that I can draw electronic circuit diagrams with Macdraw **faster than I could do freehand** with a pencil, with the added bonus of complete freedom to rearrange anything.

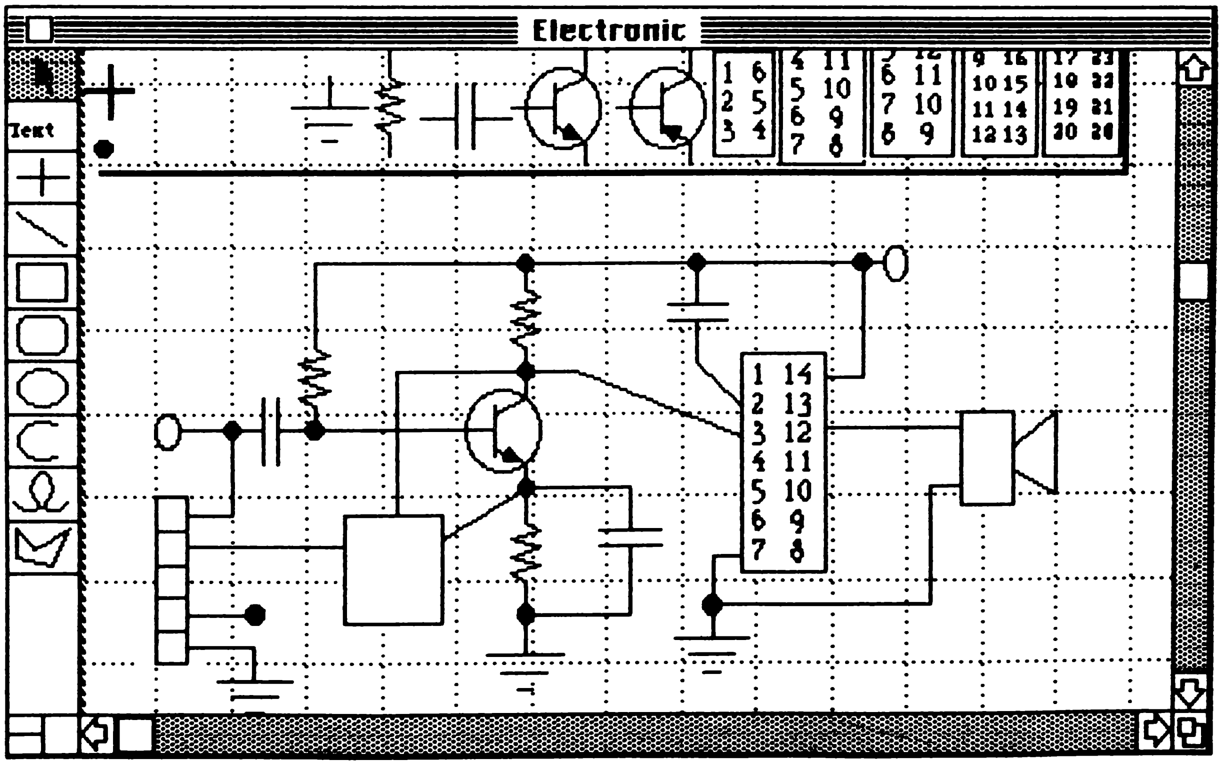
With the selection feature, any symbol or section of the circuit diagram, or even the whole circuit, can be moved around on the paper to allow more room as required as the drawing progresses.

The quality of drawing is quite acceptable, as Figure Four shows. Also there's virtually no limit to the size of the drawing: it could be up to 50 by 96 inches if required.

The drawing can be shrunk or expanded as required. It can even be printed out sideways on the paper.

Figure Four shows a direct screen dump, so it looks exactly as it would on the screen, but by selecting the **print** option from the menu the circuit diagram can be printed out without all of that stuff around the edges and also without the grid. The electronic symbols menu could also be erased before printing.

FIGURE FOUR. Using Macdraw for electronic circuit diagrams:



WINDING UP:

Yes I'm impressed. I'm wondering what software is on its way.... printed circuit board layout?.... interfacing to an X-Y plotter?.... 3-dimensional graphics?.... zooming effects?.... Macpaint has some very nice features: how about a program that blends the best of Macpaint and Macdraw? At the moment that can be done, though awkwardly, via the clipboard.

Plotter interfacing and local networking are on the way, so I've heard. Mac is so new on the market, and so different, that it's taking time for the software houses to come out with the applications programs, but the avalanche is imminent. Maybe even by the time this article appears on the news stands there'll be a whole new exciting range of software out. Go along to your local Apple dealer and find out.□

Wall Street Stimulation

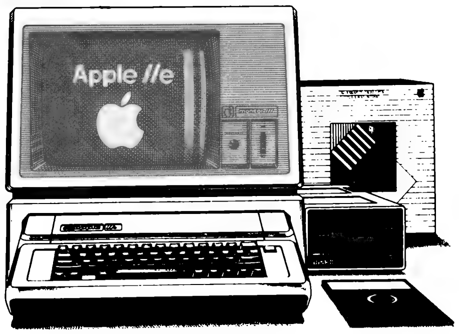
by Kevin Leong

For the Apple II computer

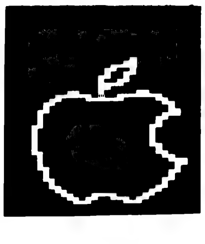
LINES 121-135 Documentation and instructions
LINES 1290-1410 Players and amount to start off
LINES 1635-1646 Company names on the sharemarket list
LINES 1660-1665 Company shares at opening
LINES 6400-6510 Company amount set to split in half

```
]LIST

6  REM
7  REM  BY KEVIN LEONG
8  REM
10  CALL  936
20  PRINT "$$$$$$$$$$$$$$$$$$$$$$
    $$$$$$$$$$$$$$$$$"
30  FOR Z = 1 TO 3: PRINT "": NEXT
 •  Z
40  PRINT "     $$$ WALL STREET ST
    IMULATION $$$     "
50  FOR Y = 1 TO 3: PRINT "": NEXT
    Y
60  PRINT "$$$$$$$$$$$$$$$$$$$$$$
    $$$$$$$$$$$$$$$$$"
70  FOR X = 1 TO 6: PRINT "": NEXT
    X
90  FOR W = 1 TO 6: PRINT "": NEXT
    W
93  PRINT "        PRESS A KEY TO
    CONTINUE           ": GET Z$
95  CALL - 936
100  INPUT "DO YOU WANT INSTRUCTI
     ONS (Y/N)?";Z$
110  IF Z$ = "N" GOTO 190
112  IF Z$ = "Y" THEN 120
115  CALL - 936: GOTO 100
120  CALL - 936
121  PRINT "YOU WILL BE SHOWN THE
      DAILY WALL STREET"
122  PRINT "STOCK MARKET REPORT.Y
     OU HAVE THE          "
123  PRINT "OPPORTUNITY TO BUY OR
      SELL SHARES.WHEN"
124  PRINT "YOU WANT TO SELL TYPE
      A HYTHEN (-)          "
125  PRINT "BEFORE A FIGURE.EG.-1
     000.AS YOU ARE     "
126  PRINT "GIVEN $5000 YOU MAY B
     UY OR SELL ANYTIME"
130  PRINT "EACH DAY THE VALUE OF
      THE SHARES "
131  PRINT "WILL FLUCTUATE AND TH
     E CHANGES WILL BE"
132  PRINT "INDICATED.COMPANIES M
     AY GO BANKRUPT AND"
133  PRINT "THEY MAY NOT RECOVER.
     THE MARKET CAN      "
134  PRINT "CAN ALSO SUDDENLY COL
     LAPSE.ONE TO FOUR "
135  PRINT "PLAYERS MAY PLAY."
140  VTAB 20: INVERSE
150  INPUT "PRESS S TO START GAME
     ";SA$
160  IF SA$ = "S" THEN 190
170  NORMAL : GOTO 120
180  PRINT "PRESS ANY LETTER TO C
     ONTINUE.": GET Y$
190  NORMAL : CALL - 936: GOSUB
     1280
200  CALL - 936: INVERSE : SPEED=
     75
203  PRINT "HERE IS THE MARKET BE
     FORE THE"
204  PRINT "COMMENCEMENT OF TRADI
     NG."
205  NORMAL : SPEED= 255
210  FOR Z = 1 TO 700: NEXT Z
220  GOSUB 1600
225 A = 0
230 A5 = B5 = C5 = D5 = E5 = F5 =
     G5 = H5 = I5 = J5 = K5 = L5 =
     0
235 A = A + 1
237  FOR CC = 1 TO 6: PRINT : NEXT
     CC
240  FLASH : PRINT "N E W S
       F L A S H !"
245  NORMAL
250 V =  INT ( RND (1) * 12)
255  GOTO 8000 •
260  IF V = 0 GOTO 400
270  IF V = 1 GOTO 410
280  IF V = 2 GOTO 420
290  IF V = 3 GOTO 430
300  IF V = 4 GOTO 440
310  IF V = 5 GOTO 450
320  IF V = 6 GOTO 460
330  IF V = 7 GOTO 470
340  IF V = 8 GOTO 480
350  IF V = 9 GOTO 490
360  IF V = 10 GOTO 500
370  IF V = 11 GOTO 380
380 X$ = A$:A5 = .9: GOTO 520
385 A5 = 0
400 X$ = B$:B5 = .9: GOTO 520
405 B5 = 0
410 X$ = C$:C5 = .9: GOTO 520
415 C5 = 0
420 X$ = D$:D5 = .9: GOTO 520
425 D5 = 0
430 X$ = E$:E5 = .9: GOTO 520
435 E5 = 0
440 X$ = F$:F5 = .9: GOTO 520
445 F5 = 0
450 X$ = G$:G5 = .9: GOTO 520
455 G5 = 0
460 X$ = H$:H5 = .9: GOTO 520
465 H5 = 0
470 X$ = I$:I5 = .9: GOTO 520
475 I5 = 0
480 X$ = J$:J5 = .9: GOTO 520
485 J5 = 0
490 X$ = K$:K5 = .9: GOTO 520
495 K5 = 0
500 X$ = L$:L5 = .9
520  PRINT X$;" AND THEIR BOARD O
     F DIRECTORS"
521  PRINT "ANNOUNCE A STRONG FIN
     ANCIAL STANDING."
530  FOR BB = 1 TO 1200: NEXT BB
540 B = 1
550  GOTO 2000
1280  FOR Z = 1 TO 13: PRINT "": NEXT
      Z
1285  CALL - 936
1290  PRINT " HOW MANY INVESTORS
      (1-4) ";
1300  INPUT X
1305  CALL - 936
1310  PRINT "INVESTOR 1 IS ";
1320  INPUT M$:MZ3 = 5000
1330  IF X   2 GOTO 1500
1340  PRINT "INVESTOR 2 IS ";
1350  INPUT N$:NZ3 = 5000
1360  IF X < 3 GOTO 1500
1370  PRINT "INVESTOR 3 IS ";
1380  INPUT O$:OZ3 = 5000
1390  IF X   4 GOTO 1500
1400  PRINT "INVESTOR 4 IS ";
1410  INPUT P$:PZ3 = 5000
1500  RETURN
1600  INVERSE : SPEED= 75: PRINT
      "MARKET AT OPENING.": NORMAL
      : SPEED= 255
1610  PRINT "------------------------
      -------------------"
1620  PRINT "STOCK PRICE CHANGE S
      HARES VALUE"
1630  PRINT "------------------------
      -------------------"
1635 A$ = "MIM"
1636 B$ = "AUG"
1637 C$ = "2SM"
1638 D$ = "CUB"
1639 E$ = "ANZ"
1640 F$ = "GWA"
1641 G$ = "DAC"
1642 H$ = "RGB"
1643 I$ = "SEX"
1644 J$ = "TNT"
1645 K$ = "SOC"
1646 L$ = "CIC"
1650 A1 = A2 = A3 = A4 = A5 = B1 =
     B2 = B3 = B4 = B5 = C1 = C2 =
     C3 = C4 = C5 = D1 = D2 = D3 =
     D4 = D5 = E1 = E2 = E3 = E4 =
     E5 = F1 = F2 = F3 = F4 = F5 =
     G1 = G2 = G3 = G4 = G5 = H1 =
     H2 = H3 = H4 = H5 = I1 = I2 =
     I3 = I4 = I5 = J1 = J2 = J3 =
     J4 = J5 = O1 = O2 = O3 = O4 =
     O5 = P1 = P2 = P3 = P4 = P5 =
     0
1660 A6 = 2.00:B6 = 2.00:C6 = 2.0
     0:D6 = 2.00:
1665 E6 = 2.00:F6 = 2.00:G6 = 2.0
     0:H6 = 2.00:I6 = 2.00:J6 = 2
     .00:K6 = 2.00:L6 = 2.00
1670  PRINT A$;"    ";A6;"      000
         000      000"
1680  PRINT B$;"    ";B6;"      000
         000      000"
1690  PRINT C$;"    ";C6;"      000
         000      000"
1700  PRINT D$;"    ";D6;"      000
         000      000"
1710  PRINT E$;"    ";E6;"      000
         000      000"
1720  PRINT F$;"    ";F6;"      000
         000      000"
1730  PRINT G$;"    ";G6;"      000
         000      000"
1740  PRINT H$;"    ";H6;"      000
         000      000"
1750  PRINT I$;"    ";I6;"      000
         000      000"
1760  PRINT J$;"    ";J6;"      000
         000      000"
1770  PRINT K$;"    ";K6;"      000
         000      000"
1780  PRINT L$;"    ";L6;"      000
         000      000"
1790  PRINT "------------------------
      -------------------"
1800  INPUT "PRESS N TO CONTINUE"
      ;Y$
1805  IF Y$ = "N" THEN 1820
1810  GOTO 1800
1820  RETURN
```

```
2000  IF B > 1 GOTO 2130
2010  GOSUB 5000
2015 A5 = A5 + T: GOSUB 5000
2020 B5 = B5 + T: GOSUB 5000
2030 C5 = C5 + T: GOSUB 5000
2040 D5 = D5 + T: GOSUB 5000
2050 E5 = E5 + T: GOSUB 5000
2060 F5 = F5 + T: GOSUB 5000
2070 G5 = G5 + T: GOSUB 5000
2080 H5 = H5 + T: GOSUB 5000
2090 I5 = I5 + T: GOSUB 5000
2100 J5 = J5 + T: GOSUB 5000
2110 K5 = K5 + T: GOSUB 5000
2120 L5 = L5 + T: GOTO 9000
2125 A6 = A6 + A5:B6 = B6 + B5:C6
      = C6 + C5:D6 = D6 + D5:E6 =
     E6 + E5:F6 = F6 + F5:G6 = G6
      + G5:H6 = H6 + H5:I6 = I6 +
     I5:J6 = J6 + J5:K6 = K6 + K5
     :L6 = L6 + L5
2128  GOTO 7000
2130  IF B = 1 THEN V$ = M$:V1 =
     M1:V2 = M2:V3 = M3:V4 = M4:V
     5 = M5:V6 = M6:V7 = M7:V8 =
     M8:V9 = M9:VVO = MVO:VW1 = M
     W1:VX2 = MX2
2133 RZ = O
2135  IF B = 1 THEN VZ3 = MZ3
2140  IF B > X GOTO 6390
2150  IF B = 2 THEN V$ = N$:V1 =
     N1:V2 = N2:V3 = N3:V4 = N4:V
     5 = N5:V6 = N6:V7 = N7:V8 =
     N8:V9 = N9:VVO = NVO:VW1 = N
     W1:VX2 = NX2
2155  IF B = 2 THEN VZ3 = NZ3
2160  IF B > X GOTO 6390
2170  IF B = 3 THEN V$ = O$:V1 =
     O1:V2 = O2:V3 = O3:V4 = O4:V
     5 = O5:V6 = O6:V7 = O7:V8 =
     O8:V9 = O9:VVO = OVO:VW1 = O
     W1:VX2 = OX2
2175  IF B = 3 THEN VZ3 = OZ3
2180  IF B   X GOTO 6390
2190  IF B = 4 THEN V$ = P$:V1 =
     P1:V2 = P2:V3 = P3:V4 = P4:V
     5 = P5:V6 = P6:V7 = P7:V8 =
     P8:V9 = P9:VVO = PVO:VW1 = P
     W1:VX2 = PX2
2195  IF B = 4 THEN VZ3 = PZ3
2200  IF B   X GOTO 6390

2210  GOTO 5140
5000 T = (( INT (40 -  INT ( RND
     (1) * 80)) / 100) * 100) / 1
     00
5010  RETURN
5140 W1 = V1 * A6:W2 = V2 * B6:W3
      = V3 * C6:W4 = V4 * D6:W5 =
     V5 * E6:W6 = V6 * F6:W7 = V7
      * G6:W8 = V8 * H6:W9 = V9 *
     I6:WVO = VVO * J6:WW1 = VW1 *
     K6:WX2 = VX2 * L6
5141  IF RZ > O GOTO 5195
5145  GOTO 9200
5150  IF DI$ = "A" GOTO 5195
5160  PRINT "****  " + DI$ + " DE
     CLARES A 10% DIVIDEND ****"
5195  INVERSE : PRINT "STOCK REPO
     RT FOR ";V$;"     DAY ";A: NORMAL

5200  PRINT "====================
     ===================="
5210  PRINT "STOCK  PRICE  CHANGE
       SHARES  VALUE"
5220  PRINT "====================
     ===================="
5225  IF A6 <  = O GOTO 6200
5230  PRINT A$; TAB( 8);A6; TAB(
     15);A5; TAB( 23);V1; TAB( 31
     );W1
5235  IF B6 <  = O GOTO 6210
5240  PRINT B$; TAB( 8);B6; TAB(
     15);B5; TAB( 23);V2; TAB( 31
     );W2
5245  IF C6 <  = O GOTO 6220
5250  PRINT C$; TAB( 8);C6; TAB(
     15);C5; TAB( 23);V3; TAB( 31
     );W3
5255  IF D6 <  = O GOTO 6230
5260  PRINT D$; TAB( 8);D6; TAB(
     15);D5; TAB( 23);V4; TAB( 31
     );W4
5265  IF E6 <  = O GOTO 6240
5270  PRINT E$; TAB( 8);E6; TAB(
     15);E5; TAB( 23);V5; TAB( 31
     );W5
5275  IF F6 <  = O GOTO 6250
5280  PRINT F$; TAB( 8);F6; TAB(
     15);F5; TAB( 23);V6; TAB( 31
     );W6
5285  IF G6 <  = O GOTO 6260
5290  PRINT G$; TAB( 8);G6; TAB(
     15);G5; TAB( 23);V7; TAB( 31
     );W7
5295  IF H6 <  = O GOTO 6270
5300  PRINT H$; TAB( 8);H6; TAB(
     15);H5; TAB( 23);V8; TAB( 31
     );W8
5305  IF I6 <  = O GOTO 6280
5310  PRINT I$; TAB( 8);I6; TAB(
     15);I5; TAB( 23);V9; TAB( 31
     );W9
5315  IF J6 <  = O GOTO 6290
5320  PRINT J$; TAB( 8);J6; TAB(
     15);J5; TAB( 23);VVO; TAB( 3
     1);WVO
5325  IF K6 <  = O GOTO 6300
5330  PRINT K$; TAB( 8);K6; TAB(
     15);K5; TAB( 23);VW1; TAB( 3
     1);WW1
5335  IF L6 <  = O GOTO 6310
5340  PRINT L$; TAB( 8);L6; TAB(
     15);L5; TAB( 23);VX2; TAB( 3
     1);WX2
5350  PRINT "====================
     ===================="
5360 VZ3 = VZ3 + DI / 10
5361 VY4 = VZ3 + W1 + W2 + W3 + W
     4 + W5 + W6 + W7 + W8 + W9 +
     WVO + WW1 + WX2
5362 DI = O
5363  IF VY4 >  = O GOTO 5370
5365  PRINT "SORRY, "V$" , BUT YO
     U ARE": PRINT "BANKRUPT AND
     ARE OUT OF THE GAME.": PRINT
     "PRESS N TO CONTINUE.":VZ3 =
      - 1000000: GOTO 5400
5370  PRINT "AVAILABLE CASH = ";V
     Z3
5380  PRINT "TOTAL ASSETS = ";VY4

5390  PRINT "WHAT COMPANY ?(PRESS
      N IF NONE);
5400  INPUT Q$
5410  IF Q$ = "N" THEN B = B + 1:
      GOTO 6100
5420  PRINT "HOW MANY SHARES ?"
5430  INPUT Q
5435 RZ = RZ + 1
5440  IF Q$ = A$ THEN VA5 = VZ3 -
     (A6 * Q): GOTO 5560
5450  IF Q$ = B$ THEN VA5 = VZ3 -
     (B6 * Q): GOTO 5590
5460  IF Q$ = C$ THEN VA5 = VZ3 -
     (C6 * Q): GOTO 5620
5470  IF Q$ = D$ THEN VA5 = VZ3 -
     (D6 * Q): GOTO 5650
5480  IF Q$ = E$ THEN VA5 = VZ3 -
     (E6 * Q): GOTO 5680
5490  IF Q$ = F$ THEN VA5 = VZ3 -
     (F6 * Q): GOTO 5710
5500  IF Q$ = G$ THEN VA5 = VZ3 -
     (G6 * Q): GOTO 5740
5510  IF Q$ = H$ THEN VA5 = VZ3 -
     (H6 * Q): GOTO 5770
5520  IF Q$ = I$ THEN VA5 = VZ3 -
     (I6 * Q): GOTO 5800
5530  IF Q$ = J$ THEN VA5 = VZ3 -
     (J6 * Q): GOTO 5830
5540  IF Q$ = K$ THEN VA5 = VZ3 -
     (K6 * Q): GOTO 5860
5550  IF Q$ = L$ THEN VA5 = VZ3 -
     (L6 * Q): GOTO 5890
5555  GOTO 6000
5560  IF VA5 < O GOTO 6000
5570  IF V1 + Q < O GOTO 6000
5575  IF A6 < O GOTO 6000
5580 V1 = V1 + Q: GOTO 5990
5590  IF VA5 < O GOTO 6000
 600  IF V2 + Q < O GOTO 6000
 605  IF B6 < O GOTO 6000
 610 V2 = V2 + Q: GOTO 5990
 620  IF VA5   O GOTO 6000
5630  IF V3 + Q < O GOTO 6000
5635  IF C6 < O GOTO 6000
5640 V3 = V3 + Q: GOTO 5990
5650  IF VA5 < O GOTO 6000
5660  IF V4 + Q < O GOTO 6000
5665  IF D6 < O GOTO 6000
5670 V4 = V4 + Q: GOTO 5990
5680  IF VA5 < O GOTO 6000
5690  IF V5 + Q < O GOTO 6000
5695  IF E6 < O GOTO 6000
5700 V5 = V5 + Q: GOTO 5990
5710  IF VA5 < O GOTO 6000
5720  IF V6 + Q < O GOTO 6000
5725  IF F6 < O GOTO 6000
5730 V6 = V6 + Q: GOTO 5990
5740  IF VA5 < O GOTO 6000
5750  IF V7 + Q < O GOTO 6000
5755  IF G6   O GOTO 6000
5760 V7 = V7 + Q: GOTO 5990
5770  IF VA5 < O GOTO 6000
5780  IF V8 + Q < O GOTO 6000
5785  IF H6 < O GOTO 6000
5790 V8 = V8 + Q: GOTO 5990
5800  IF VA5 < O GOTO 6000
5810  IF V9 + Q < O GOTO 6000
5815  IF I6 < O GOTO 6000
5820 V9 = V9 + Q: GOTO 5990
5830  IF VA5 < O GOTO 6000
5840  IF VVO + Q < O GOTO 6000
5845  IF J6 < O GOTO 6000
5850 VVO = VVO + Q: GOTO 5990
5860  IF VA5 < O GOTO 6000
5870  IF VW1 + Q < O GOTO 6000
5875  IF K6 < O GOTO 6000
5880 VW1 = VW1 + Q: GOTO 5990
5890  IF VA5 < O GOTO 6000
5900  IF VX2 + Q < O GOTO 6000
5905  IF L6 < O GOTO 6000
5910 VX2 = VX2 + Q: GOTO 5990
5990 VZ3 = VA5: HOME : GOTO 5140
6000  PRINT "TRY AGAIN .THINK THI
      S TIME!!!!!": GOTO 5140
6100  IF V$ = M$ THEN M1 = V1:M2 =
      V2:M3 = V3:M4 = V4:M5 = V5:M
```



```
       6 = V6:M7 = V7:M8 = V8:M9 =
       V9:MVO = VVO:MW1 = VW1:MX2 =
       VX2:MZ3 = VZ3
6110  IF V$ = N$ THEN N1 = V1:N2 =
       V2:N3 = V3:N4 = V4:N5 = V5:N
       6 = V6:N7 = V7:N8 = V8:N9 =
       V9:NVO = VVO:NW1 = VW1:NX2 =
       VX2:NZ3 = VZ3
6120  IF V$ = O$ THEN O1 = V1:O2 =
       V2:O3 = V3:O4 = V4:O5 = V5:O
       6 = V6:O7 = V7:O8 = V8:O9 =
       V9:OVO = VVO:OW1 = VW1:OX2 =
       VX2:OZ3 = VZ3
6130  IF V$ = P$ THEN P1 = V1:P2 =
       V2:P3 = V3:P4 = V4:P5 = V5:P
       6 = V6:P7 = V7:P8 = V8:P9 =
       V9:PVO = VVO:PW1 = VW1:PX2 =
       VX2:PZ3 = VZ3
6135 RZ = 0
6140  GOTO 2130
6200  PRINT A$;" IS BANKRUPT":A6 =
       - .7:V1 = 0:W1 = 0: GOTO 52
       35
6210  PRINT B$;" IS BANKRUPT":B6 =
       - .7:V2 = 0:W2 = 0: GOTO 52
       45
6220  PRINT C$;" IS BANKRUPT":C6 =
       - .7:V3 = 0:W3 = 0: GOTO 52
       55
6230  PRINT D$;" IS BANKRUPT":D6 =
       - .7:V4 = 0:W4 = 0: GOTO 52
       65
6240  PRINT E$;" IS BANKRUPT":E6 =
       - .7:V5 = 0:W5 = 0: GOTO 52
       75
6250  PRINT F$;" IS BANKRUPT":F6 =
       - .7:V6 = 0:W6 = 0: GOTO 52
       85
6260  PRINT G$;" IS BANKRUPT":G6 =
       - .7:V7 = 0:W7 = 0: GOTO 52
       95
6270  PRINT H$;" IS BANKRUPT":H6 =
       - .7:V8 = 0:W8 = 0: GOTO 53
       05
6280  PRINT I$;" IS BANKRUPT":I6 =
       - .7:V9 = 0:W9 = 0: GOTO 53
       15
6290  PRINT J$;" IS BANKRUPT":J6 =
       - .7:VVO = 0:WVO = 0: GOTO
       5325
6300  PRINT K$;" IS BANKRUPT":K6 =
       - .7:VW1 = 0:WW1 = 0: GOTO
       5335
6310  PRINT L$;" IS BANKRUPT":L6 =
       - .7:VX2 = 0:WX2 = 0: GOTO
       5350
6390  FLASH : CALL  - 936
6400  IF A6 > 5.8 THEN M1 = 2 * M
       1:N1 = 2 * N1:O1 = O1 * 2:P1
        = P1 * 2:A6 = 3.10 - L5: PRINT
       A$;"       HAS SPLIT ITS SHAR
       ES 2 FOR  1."
6410  IF B6 > 6.3 THEN M2 = M2 *
       2:N2 = 2 * N2:O3 = 2 * O3:P3
        = 2 * P3:B6 = 3.10 - L5: PRINT
       B$;"      HAS SPLIT ITS SHARE
       S 2 FOR 1."
6420  IF C6 > 5.9 THEN M3 = 2 * M
       3:N3 = 2 * N3:O3 = 2 * O3:P3
        = 2 * P3:C6 = 3.00 + L5: PRINT
       C$;"       HAS SPLIT ITS SHAR
       ES 2 FOR 1."
6430  IF D6 > 6.1 THEN M4 = 2 * M
       4:N4 = 2 * N4:O4 = 2 * O4:P4
        = 2 * P4:D6 = 3.20 - L5: PRINT
       D$;"       HAS SPLIT ITS SHAR
       ES 2 FOR 1."
6440  IF E6 > 6.4 THEN M5 = 2 * M
       5:N5 = 2 * N5:O5 = 2 * O5:P5
        = 2 * P5:E6 = 3.00 - L5: PRINT
       E$;"       HAS SPLIT ITS SHAR
       ES 2 FOR 1."
6450  IF F6 > 6 THEN M6 = 2 * M6:
       N6 = 2 * N6:O6 = 2 * O6:P6 =
       2 * P6:F6 = 3.50   L5: PRINT
       F$;"  HAS SPLIT ITS SHARES 2
        FOR 1."
6460  IF G6   6 THEN M7 =   * M7:
       N7 = 2 * N7:O7 = 2 * O7:P7 =
       2 * P7:G6 = 2.90 + L5: PRINT
       G$;"  HAS SPLIT ITS SHARES 2
        FOR 1."
6470  IF H6 > 6 THEN M8 = 2 * M8:
       N8 = 2 * N8:O8 = 2 * O8:P8 =
       2 * P8:H6 = 3.30 - L5: PRINT
       H$;"  HAS SPLIT ITS SHARES 2
        FOR 1."
6480  IF I6 > 5.7 THEN M9   2 * M
       9:N9 = 2 * N9:O9 = 2 * O9:P9
        = 2 * P9:I6 = 3.00   L5: PRINT
       I$;"  HAS SPLIT ITS SHARES 2
        FOR 1."
6490 . IF J6    6.3 THEN MVO = 2 *
       MVO:NVO = 2 * NVO:OVO = 2 *
       OVO:PVO = 2 * PVO:J6 = 3.00 +
       L5: PRINT J$;"  HAS SPLIT IT
       S SHARES 2 FOR 1."
6500  IF K6 > 6 THEN MW1 = 2 * MW
       1:NW1 = 2 * NW1:OW1 = 2 * OW
       1:PW1 = 2 * PW1:K6 = 2.99 +
       L5: PRINT K$;"  HAS SPLIT IT
       S SHARES 2 FOR 1."
6510  IF L6 > 6 THEN MX2 = 2 * MX
       2:NX2 = 2 * NX2:OX2 = 2 * OX
       2:PX2 = 2 * PX2:L6 = 3.50 -
       L5: PRINT L$;"  HAS SPLIT IT
       S SHARES 2 FOR 1."
6515  FOR DD = 1 TO 6: PRINT : NEXT
       DD
6520 BB =  RND (1)
6530  IF BB > .04 GOTO 6560
6540  PRINT "MARKET CRASH: ALL PR
       ICES HALVED!!!"
6550 A6 = A6 / 2:B6 = B6 / 2:C6 =
       C6 / 2:D6 = D6 / 2:E6 = E6 /
       2:F6 = F6 / 2:G6 = G6 / 2:H6
        = H6 / 2:I6 = I6 / 2:J6 = J
       6 / 2:K6 = K6 / 2:L6 = L6 /
       2
6560 SH =  INT ( RND (1) * 30) +
       1
6570  GOTO 230
7000 A6 =  INT (A6 * 100) / 100
7010 B6 =  INT (B6 * 100) / 100
7020 C6 =  INT (C6 * 100) / 100
7030 D6 =  INT (D6 * 100) / 100
7040 E6 =  INT (E6 * 100) / 100
7050 F6 =  INT (F6 * 100) / 100
7060 G6 =  INT (G6 *.100) / 100
7070 H6 =  INT (H6 * 100) / 100
7080 I6 =  INT (I6 * 100) / 100
7090 J6 =  INT (J6 * 100) / 100
7100 K6 =  INT (K6 * 100) / 100
7110 L6 =  INT (L6 * 100) / 100
7120  GOTO 2130
8000 A5 = 0
8010 B5 = 0
8020 C5 = 0
8030 D5 = 0
8040 E5 = 0
8050 F5 = 0
8060 G5 = 0
8070 H5 = 0
8080 I5 = 0
8090 J5 = 0
8100 K5 = 0
8110 L5 = 0
8120  GOTO 260
9000 A5 =  INT (A5 * 100) / 100
9010 B5 =  INT (B5 * 100) / 100
9020 C5 =  INT (C5 * 100) / 100
9030 D5 =  INT (D5 * 100) / 100
9040 E5 =  INT (E5 * 100) / 100
9050 F5 =  INT (F5 * 100) / 100
9060 G5 =  INT (G5 * 100) / 100
9070 H5 =  INT (H5 * 100) / 100
9080 I5 =  INT (I5 * 100) / 100
9090 J5 =  INT (J5 * 100) / 100
9100 K5 =  INT (K5 * 100) / 100
9110 L5 =  INT (L5 * 100) / 100
9120  GOTO 2125
9200  IF SH = 1 THEN DI$ = "MIM":
       DI = W1
9210  IF SH = 2 THEN DI$ = "AUG":
       DI = W2
9220  IF SH = 3 THEN DI$ = "2SM":
       DI = W3
9230  IF SH = 4 THEN DI$ = "CUB":
       DI = W4
9240  IF SH = 5 THEN DI$ = "ANZ":
       DI = W5
9250  IF SH = 6 THEN DI$ = "GWA":
       DI = W6
9260  IF SH = 7 THEN DI$ = "DAC":
       DI = W7
9270  IF SH = 8 THEN DI$ = "RGB":
       DI = W8
9280  IF SH = 9 THEN DI$ = "SEX":
       DI = W9
9290  IF SH = 10 THEN DI$ = "TNT"
       :DI = WVO
9300  IF SH = 11 THEN DI$ = "SOC"
       :DI = WW1
9310  IF SH = 12 THEN DI$ = "CIC"
       :DI = WX2
9320  IF SH > 12 THEN DI$ = "A"
9330  GOTO 5150
```



ONERR GOTO message routine

by Lee Reynolds

Instead of having your program bomb out when an error occurs, incorporate Applesoft's powerful ONERR GOTO statement.

Some readers will have made use of the ONERR GOTO statement in their Applesoft programming. This capability of the language is very useful for what is called "trapping" errors, instead of having to put up with the alternative, which is to have the program bomb when an error occurs.

For those readers who are not acquainted with the use of this statement, here is an example: suppose one of the functions of a program was to DELETE temporary or unwanted files from a floppy disk. Presumably, the name of each file to be DELETEd would be known to the program, either because a set name was programmed in beforehand, or because the program would request the name from the operator.

Now, what happens during the execution of this program if (1) you get a read error on the disk, (2) the file was not found in the catalog, or (3) the file was there, but it had been LOCKed?

Without an ONERR GOTO routine, the program will "bomb" and you will be returned to Applesoft after the appropriate DOS error message has been displayed.

If the delete command is preceded by the statement:
1ØØ ONERR GOTO 25ØØØ
then an error condition will cause the program to GOTO line 25ØØØ instead of bombing. In this example, line 25Ø ØØ is the beginning of the "error trapping" routine.

Type of error

One of the first things your program will want to do at line 25ØØØ is to determine what type of error occurred. This can be determined by examining the contents of memory location 222 with the PEEK function. The error handling routine may also be used to determine the line number at which the error occurred; the guilty line is given by the expression LINE=PEEK (218)=PEEK (219)*256. (The same error could happen in different parts of a program, and the manner by which it is handled may be dependent on the line number where it occurred.)

The three different cases alluded to in the opening paragraph above can be identified by PEEK (222) having three different values: 8, 6 and 1Ø. An IF statement could test each of these cases and, when true, take appropriate action. For example, an I/O error in case (1), might be handled by simple PRINTing some statement and then STOPping.

In case (2) of FILE NOT FOUND, a message could read "File not on this disk. Please insert proper disk and hit return", then when the operator had followed instructions, the program would go back and try to delete the program again.

In case (3) of FILE NOT FOUND, you might display a message like "File locked. Do you want it developed? (Y/N)", and then, depending on the operator's answer, do whatever is necessary to unlock and delete it, or continue.

Other error

If some other error than the above three cases had caused your program to GOTO line 25ØØØ, then the value of PEEK (222) would be something other than 6, 8 or 1Ø. The value returned depends on the error encountered.

Program listing 1

```
25000  REM ERROR TRAPPING ROUTINE
25010  ERRNO = PEEK (222) : LINE =
      PEEK (218) + PEEK (219) * 256 :
      ASADDR = 53856 : DOSADDR = 43380
25100  ADDR = ASADDR + ERRNO :  IF
      ERRNO > 0 AND ERRNO < 16 THEN
      GOTO 25130
25110  PRINT CHR$(PEEK(ADR));:  ADDR
      = ADDR + 1 : IF PEEK (ADDR) <
      192 THEN GOTO 25110
25120  PRINT CHR$(PEEK(ADDR)-128) :
      STOP
25130  ADDR = DOSADDR : IF ERRNO = 1
      THEN GOTO 25110
25140  N1=0 : N2 = ERRNO-2 : IF ERRNO
      < 4 THEN N2 = 1
25150  ADDR = ADDR + 1 : IF PEEK
      (ADDR) < 192 THEN GOTO 25150
25160  N1 = N1 + 1 : IF N1 < N2 THEN
      GOTO 25150
25170  ADDR = ADDR + 1 : GOTO 25110
```

▷

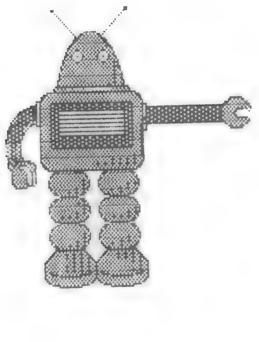
There are two general categories to consider: (1) other DOS errors like "Write Protected" or (2) Applesoft errors, like "***Syntax Error" or "Redimensioned Array". You can find which values of PEEK (222) correspond to which error conditions by reading page 81 of the Applesoft manual or pages 114-122 of the DOS manual.

There are a total of 15 possible DOS errors and 17 Applesoft errors. It is not usually necessary for an "error trapping" routine to have to test for all 32 possibilities, but instead just to display the appropriate message and then STOP.

If you had an IF statement that tested for all 32 errors and printed a message similar to the Applesoft or DOS message, that would be a lot of wasted code. It would also be unnecessarily space-consuming, for you would be duplicating the message already in the computer. Therefore, the routine listed may be of value. It PEEKs at the locations DOS or Applesoft use to store these messages, and then PRINTs the needed message one character at a time (although it prints so fast that the entire message appears at once).

The starting address for the first Applesoft Error Message is set in the variable ASADDR by line number 25Ø1Ø. This value only applies to Applesoft in ROM (and Apple II Plus or an Applesoft BASIC card). The same line in the accompanying listing also sets the address for the first DOS error message, in the variable DOSADDR. This value is for DOS 3.2 or 3.3 on the 48k machine.

If you want your error-trapping routine to process certain errors in particular ways (not by just PRINTing the message and STOPping), then the appropriate logic could be inserted between lines 25Ø1Ø and 251ØØ. □



A glossary of custom characters

by Denis J Brothers

Generation of special text characters is a useful feature especially where scientific and other symbols are required. The Pkaso printer interface card can be used to create these characters.

Among functions such as graphics dumping, the Pkaso parallel interface card from Interactive Structures Inc permits the generation of special text characters with the Apple II and Apple //e and a dot matrix printer. This is most useful for those situations where special symbols not present in the printer's character sets are needed, eg for many scientific applications.

The Pkaso software (revision 6.1, February 1983 and later) has provision for generating a glossary file for Apple Writer II and, bit confusing this, Apple Writer //, enabling the special characters to be called up and entered at will when creating text files. Unfortunately, the manual is uninformative on the detailed procedure to be followed, and it takes quite a bit of trial and error before success is realised (at least for an inexperienced user such as myself).

Instructions

The following instructions and notes are presented to ease the way of others who may be faced with this problem. They are for Apple Writer // (for the Apple //e) and a system with only one disk drive; diskette swopping can be reduced if two drives are available. Control characters are indicated following the convention used in the Apple Writer //e manual, eg CTRL L is shown as [L].

(Incidentally, Apple Writer // is a much improved version of Apple Writer II; the similarity in the names is unfortunate because accounts of the older version are of marginal relevance for the new one.)

1. Initialise a diskette for storing the glossary.
2. Boot the PKASO diskette fully.
3. Choose the EDIT OR CREATE SPECIAL CHARACTERS option (ie, enter "E"). If using an Apple //e make sure that the CAPS LOCK key is down. Wait while NULLCHAR (the base blank character set) is loaded.
4. Reject the offer of more instructions (they are not informative for our purposes, anyway) by pressing "N".
5. Make sure that the following specifications listed are correct and change where necessary (enter "C" and follow menu): character height (7 or 8 bit) and reference system (by letter or number). Other specifications may be left in the default condition.
6. Choose the EDIT CHARS option (enter "E") and follow the instructions shown to create the characters required. The following sequence is useful: Enter the letter (or number) reference for the character, create the character by using the designated keys for moving or drawing with the cursor, press "RETURN" and wait until the screen is set up for the next character, press "ESC", choose HI-RES DISPLAY option (enter "H"), enter reference letter of character when screen is evenly lit, press "RETURN" to preview character, press "ESC", enter "E" to edit the character or create another character, etc. Previewing a character in this way is useful because the final form is horizontally compressed as compared with the impression gained during the creation process.
7. Remove PKASO diskette from

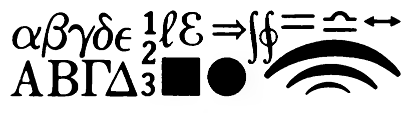
drive and replace with initialised diskette.
8. Return to PKASO CHAR DESIGN menu (press "ESC") and choose APPLE WRITER option (enter "A").
9. Make sure that the following specifications displayed are correct: printer type (default varies for different cards and will be correct if the appropriate one for your printer is fitted) and output format (Apple II – applies to //e). The other specifications are carried over from step 5 and left in the default condition.
1Ø. Choose NEW FILE option (enter "N").
11. Enter "Y" if starting a new glossary file, "N" if adding to an already existing file.
12. Name the file (eg GLOSSARY), press "RETURN".
13. Choose ENTER CHARS option (enter "E").
14. Enter each character in turn by typing its reference letter or number and pressing "RETURN". There is a brief delay while this operation is carried out.
15. Return to APPLE WRITER GLOSSARY FILE menu (press "ESC").
16. Choose DOS COMMANDS option (enter "D").
17. SAVE the file (enter "S"), choosing the appropriate options from the menu.
18. Remove the GLOSSARY diskette from the drive.
19. Boot the APPLE WRITER // MASTER diskette fully, then remove it.
2Ø. Insert the GLOSSARY diskette and load the file as a normal text file (ie, enter [L] (CTRL L) and supply the name of the glossary file).
21. Remove the GLOSSARY diskette and replace it with the APPLE WRITER // MASTER diskette but do not boot it.
22. For each entry in the glossary file displayed, position the cursor on the first control or escape character, ie, the second character of the entry. Enter [L] and load file CONTROLV (from the Apple Writer master). Position the cursor immediately after the last character in the entry and again [L]:CONTROLV. (This inserts the necessary CTRL V before and after each entry. If this is not done, control and escape characters will be lost when entering a special character from the glossary file into text.)
23. When finished with all entries, remove the APPLE WRITER // MASTER diskette and replace it with the GLOSSARY diskette.
24. SAVE the glossary file as for any other text file ([S]: Name) using the same name as given in step 12. (This replaces the first version of the file with the second which has the CTRL V characters inserted.)
25. The special character glossary file can now be used in the same way as any other glossary. When editing or entering text, load the file with [Q], E:Name and access it with [G]: reference letter.

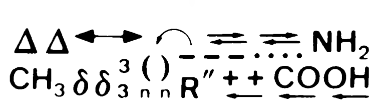
As the instructions to the printer to form a special character are much longer than the character itself, the position of the right margin may have to be increased, by inserting a .RM command at the end of the line immediately before that in which the character appears, and then decreased at the end of the line containing the character. This unfortunately means that full justification is lost (because RETURNs) have to be used before and after the .RM commands). The positions where such commands must be inserted can be found by printing to the screen or doing a test print.

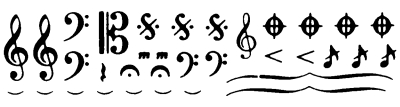
Minor peculiarity

A peculiarity (the reason for which is a mystery to me) is that if one attempts to leave leading or trailing blank bytes when generating characters so that such characters can be printed sequentially without merging, say in a special italic character set, each character is printed with a little garbage (tight squiggles) at its right end, at least when using the C.Itoh 851ØA printer. Interestingly enough, this happens only if the characters have been saved in an Apple Writer glossary file, and not if they are printed directly from the Pkaso card. Such leading or trailing blanks should thus not be used when constructing a glossary.

Furthermore, if one attempts to print any special characters sequentially, without spaces between them, the various escape and control characters sometimes seem to become confused so that the required characters may not be printed, or the printer may even change character size or line spacing.







This sort of interaction may be eliminated, and the merging of characters may be prevented, by inserting micro-spaces between them. The C.Itoh 851Ø printer permits this in proportional mode by the use of ESC followed by a number from 1 to 6, specifying the number of dot spaces.

If a special character is always likely to be used adjacent to another, it is probably a good idea to insert the necessary ESC sequences in the glossary file when editing it and inserting the CTRL Vs. If printing normal pica, for example, one would insert ESC P2 ESC N at the end of each character (before the [V] if a 2-dot space is needed).

Pkaso instructions

The Pkaso instructions state that [I]nF sequences should be used for specifying print size, but this requirement may be removed by starting the text file with [I]255F, after which the normal ESC sequences work and take up less space in the text. (Sometimes this "switching off" command does not seem to be necessary, but I have not been able to pinpoint why.)

Even if using an Apple //e, only capital letters can be used as reference letters during the creation of special characters. When the CTRL Vs are being inserted the file can be edited however, and the reference letters can be changed to lower case (or any other characters for that matter). By creating different glossary files and adding these to the first, a file of 99 special characters referenced by all available keyboard characters may be constructed.

The above account is based on my experience with the revision 6.1 Pkaso as used with an Apple //e and a C.Itoh 851ØA printer. It may be that some aspects differ for other combinations of hardware. □

The emperor has no clothes

by Gareth Powell

One of the great myths of computing is that both computers and software are easy to use. It just is not true.

In our office we have as much computer sophistication as you are likely to see in any general office.

We work with computers.

We write about computers.

We play with computers.

We repair computers.

And we have an unending stream of software coming into the office for evaluation. Indeed in many cases we are writing software ourselves.

Tough learning curve

And yet every time we start using a new piece of office software we realize we have to go through the tedious and sometimes long process of a learning curve. In some cases the learning curve takes far longer than is acceptable.

Wang Computers have set an arbitrary limit of 3Ø minutes to the time that somebody can sit down at one of their machines and learn to operate the word processor. We think 3Ø minutes is far too long.

We believe that anybody should be able to sit down at a word processor and start inputting information within ten minutes. After all, it is possible with a typewriter.

Why should it not be possible with a computer?

Accepted fact

Unfortunately, we have come to accept as a fact of life that software must be complicated, software must be difficult to learn. Software can be given the title "user friendly" when it is about as friendly as a rabid rattlesnake.

If anyone wishes to make a case for dBase II or Wordstar being user friendly we would like to hear from them.

One of the problems is the people involved with computers tend to be hackers rather than users.

Much software has a tendency to give the user an inferiority complex. Users feel if they cannot make the software perform as suggested in the advertising, there is something seriously wrong with them. And it is simply not so. The fault is nearly always either with the software or the advertising agency which has produced misleading advertisements.

Advertising hype

For example, in advertisements in the American press, MicroSoft are suggesting that with a new enhancement to MultiPlan you can move from scratch to a spreadsheet in five minutes. It simply is not so.

Although MultiPlan may be one of the best spreadsheet systems ever put together, it requires a lot of learning, a lot of patient application, a lot of understanding. It is not something you can switch on and use straight away. Even if you know it like the back of your hand, the suggestion that a pre-programmed template is going to give you a completed spreadsheet in five minutes does not stand up to logical examination.

Lip service

Within the software industry a large percentage of producers already pay lip service to the need for easy-to-use software. But it is only lip service.

We always consider it a sign of defeat if a program is dotted throughout with help screens. If the program is well-written then why should it require all those help screens?

And why is it the newspapers are full of advertisements for companies who offer tutorials on how to use programs which are supposed to be relatively user friendly?

The major problem lies in the fact that we have no standard.

There is no standard way to lay out a menu.

There is no standard way to work a spreadsheet.

There is no standard way to operate a word processing system.

Undoubtedly we are going to move in that direction with the integrated programs coming on the market.

Inverse law

The second reason why software is very difficult for users to operate is there is a law of inverse proportion.

The easier it is for software to be used the harder it is to create.

And the harder the software is to use then probably the easier it was to write in the first place. Sloppy writing leads to hard-to-use software. To make software work well for a user, a large amount of the program must be devoted simply to the user interface.

Another reason why some software is difficult to use is that some of the potential buyers and many of the programmers have been brought up in data processing departments.

Although this is not a terminal disease – pun unintentional – it can make life very difficult. Such people believe if a program does not look as though it will run on a mainframe; if a program has documentation that is slim and easy to understand; if a program boots and operates first time – then that program is not worth buying.

Over buying

It is interesting that where data processing departments are involved in buying programs for personal computers within a company, they nearly always buy programs immensely more powerful than are needed.

These are the same people who waited until IBM put their initials on the front of a computer before they believed that the personal computer age had arrived.

Software is normally not very easy to use because a lot of the people involved in putting it together do not care enough.

Programmers – they are the men with beards and corduroy trousers and the packets of boiled lollies in their pockets – do not care if a program is easy to use.

They care whether it can do as much as possible within the space available.

They write in more and more features and forget that in doing so they are making it more and more difficult to understand and cope with.

Communication problems

By definition programmers are technically-minded people. and technically-minded people have great difficulty in communicating to the large mass of humanity that have to use programs.

Programmers love to add fiddly bits onto programs that are simply not needed but make the programmer feel he has written a more elegant program.

And we as software reviewers do not actually help.

We have a fairly wide experience with software and so we can make educated guesses as to how the software will work. Because of this, documentation which is even more than half reasonable is considered splendid.

And documentation which would be thrown out by an Albanian engine manufacturer is accepted as reasonable.

Dealer problems

Near the end of the line the dealer has to cope with the software, and the dealer – if he has any sense and most dealers are very wise in this way – restricts the number of programs he stocks to those he understands.

The result is the number of programs available in any shop in Australia rarely exceeds 5Ø. And the average would be closer to 2Ø.

And quite right too.

If a dealer has to demonstrate software he does not clearly understand he is going to have one hell of a job selling it.

The poor user

At the very end of the line we get the user, who is given a demonstration to show how simple the program is to operate. Unless the user is blessed with ten gigabytes of memory, at the end of the demonstration he has forgotten most of what has been shown.

And has to depend on the documentation.

The user takes the software home or to the office, reads the documentation and immediately gets lost.

Familiar objects

What we need to see are programs which are written around familiar ways of working and familiar objects.

This is beginning to happen in the Lisa range and the Macintosh where there are icons (a daft name for small drawings) to show what you are doing.

What is needed is consistency in the menus. Consistency in the implementations of various commands. Consistency in the way in which software is written. Consistency in the way in which a program progresses.

Great hopes.

Hopefully, the industry is beginning to realise this is essential. Especially if all those people out there who have not yet even tried a personal computer are going to be persuaded to use one.

There is very little chance of getting a user to make a major decision to buy a computer if the software shown is almost unintelligible.

We can see the day arriving when the amount of documentation needed for a software program is thinned down to a very slim volume indeed.

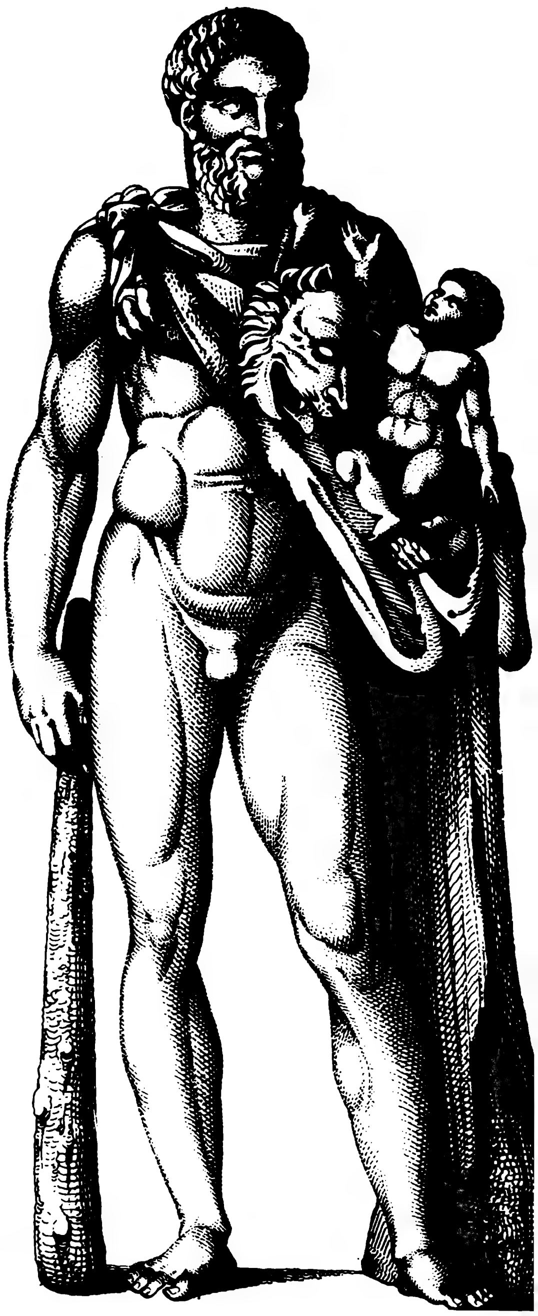
And hopefully it will have an index as a guide for moments of distress.

Hopefully we will see the day when sitting down and using a computer will be as natural a function as picking up the telephone and speaking into it.

Hopefully we will see the day when programmers realise the end user is by far the most important person.

Hopefully.

One day. □



The first Computer Of The Year that won't be out of date by next year.

The highly respected "Your Computer" magazine has just named the Apple Lisa® as Computer Of The Year, 1984.

In their own words, "People will remember 1983 as the year that Lisa revolutionised personal computing."

Surely good news for the business on the verge of choosing the ideal system.

In the frantic, fast-moving world of micro

technology, where new models are here today and gone this afternoon, Lisa seems to be a reassuring exception.

This is the most advanced personal computer in the world, with up to one million bytes of internal memory.

Unlike conventional computers, Lisa works visually, the way you do. Those complex computer commands are replaced with familiar symbols and a palm-sized mouse.

Countless man-hours are saved because Lisa starts being productive from the moment it's switched on. (Even for staff who've never used a computer before.)

Little technical miracles like these don't exactly happen overnight.

Considering they've taken us a good five years to perfect, even if our competitors simply copy, they should be kept busy for some time.

There are three Lisa models of varying price and capacity, any one of which your Apple dealer would be proud to demonstrate.

You probably won't be the only company moving to Lisa technology this year.

We expect quite a few of our competitors will be doing likewise.

apple®

Lisa. The personal computer for the office.

ENDS – utility program for Applesoft Basic

by Owen Podger

ENDS is a utility program which I use as the front and back of most programs that I write in Applesoft Basic. It consists of helpful aids to writing and running a typical interactive program: bomb-proof editable input, standard yes/no routine, and a menu generator, and a choice of the following aids during writing: RUN, conversions of hex, dec and ASCII code, list-printing, SAVEing, and RUN HELLOing.

Each part is an adaptation or improvement of published programs. It occupies 22 sectors of disk-space, or 16 sectors when remarks are removed.

FRONT END is designed to aid running of any interactive program, but the usefulness of BACK-END dies when the program is in operative order, and may then be deleted with minor revisions. ENDS then occupies 10 sectors.

This article gives details of how to use ENDS, and describes the program in detail.

FRONT END functions

INPUT is a bomb-proof input routine along the lines of Cheeseman's program (see reference), but somewhat enhanced.

In order to use INPUT, I call GOSUB 32. I may also define Q$, the leading question, IL the maximum input length (as less than the default 256), and IX=1 to restrict input to numeric mode. When I wish to "edit" rather than rewrite input, I define IR$ the right-side of the cursor. To try this out see the example at line 1000.

If the cursor is too low on the screen, the screen scrolls to allow the full length of the input to appear before the bottom line. If the cursor is higher than row 12, it is moved down to leave room for printing instructions. The leading question is screened, and with just the slightest time-delay, you may begin entering input of any key except control characters (ASCII 32).

Control characters have the following effects:

ESC : editing instructions are printed at the top of the screen (see Fig 2).

Back-arrow or ctrl-H: back space one character, but not beyond first character.

Forward-arrow or ctrl-V: forward space one character, but no further than the first space after the current input string.

Ctrl-B: return cursor to first character.

Ctrl-C: STOP program after Y/N check.

Ctrl-D: deletes character under cursor.

Ctrl-E: the cursor is returned to the first space after the current input string. If IR$ is predefined, the cursor moves to the end of the predefined string.

Ctrl-I: insert mode is initiated, and all characters will be inserted to the left of the cursor until another ctrl character is pressed.

RETURN or ctrl-M: return with input to the left of the cursor only.

Ctrl-P: return with the whole input string regardless of the location of the cursor.

When the string is four characters short of the maximum length, a bell warning rings. Input longer than the maximum cannot appear. The bell warning does not operate in numeric mode.

The INPUT subroutine also returns a pseudo-random number IR.

YES/NO HANDLER (GOSUB 71) takes a pre-defined leading question Q$, adds "Y/N?", and prompts a response. It will only accept Y or N, any other response causing the question to be repeated in INVERSE mode. The subroutine returns a true or false value to the variable YES. Two illustrations of the subroutine are used in ENDS. The first, in line 57, uses IF YES THEN. The second in lines 63930-63933 uses a FOR YES=0 TO 1 "truth-loop", which repeats until YES is true.

For MENU-GENERATOR (GOSUB 76), the main program must define OP, the number of options, a maximum of nine, and Q$(1) to Q$(OP), the name for each option, as illustrated in line 63810 and Fig 1. The subroutine displays a menu beginning "SELECT FROM THE FOLLOWING", then lists the options in order. Selection is by a key-press. The variable YES is equal to the selected option, and asterisks appear adjacent to it. On return, I use ON YES GOTO or ON YES GOSUB.

BACK-END functions

When the program is run, the list of BACK-END options is displayed on the screen. A single numeric keystroke is used to select an option.

RUN is a simple branch to the main program fron line 63812.

HEX TO DEC is found at line 63900. The prompt "HEX" is printed. When you enter a hex number and press RETURN, the decimal equivalent is printed with another prompt for a hex number. The loop is broken by entering zero.

DEC TO HEX, at line 63910, calls for a "DEC" number. Only integers work, of course. Again, the loop is broken by entering zero.

PRINT LISTING, at line 63920, gives you a paper copy of the listing, either normally or formatted to Anderson et al's LISZTER.

SAVE, at lines 63930 to 63934, will call for a program name or "CATALOG". There are three options, to enter a program name, to enter "CATALOG" or press RETURN without a name, which will SAVE with the old name as described below. You are then asked to enter drive-number. Again three options, 1, 2 or RETURN, the last not changing the drive from that last used.

Your program will not be saved under the name you give it, rather the letter A will be prefixed to it. If you RETURN without a name, the prefix will be advanced one letter, such that the program first entered as "ENDS" will be saved as "AENDS" then on each successive SAVE without entering a name, as "BENDS", "CENDS", and so on.

Entering "CATALOG" will do just that, and on a key-press you are returned to the BACK-END menu.

RUN HELLO does that at line 63940. You may need to adjust the name to suit yourself.

KEY TO ASC, at line 63950, prompts a key-press, upon which the ASC value is given, with a fresh prompt. But if RETURN is pressed the number 13 is printed and the program returns to the BACK-END menu.

STOP does that.

The final lines of BACK-END handle errors, listing the line with the error.

Use of variable

FRONT-END uses some conventions that should be noted. Variables beginning with A, B or C, followed by a digit, define commonly-used integers. B0$, B1$ and BL$ are blank strings; and D$, as usual, contains ctrl-D, with D0$ being the bell. These are defined in an initialising subroutine at lines 82-96, where arrays should be defined if appropriate.

All FRONT-END variables begin with the letter I, for Input, but the following variables communicate between FRONT-END and the rest of the program:

I numeric value of input (VAL(I$))
I$ input string
IL maximum length of input, with maximum and default being 256 less length of Q$
IR pseudo-random integer. Each time GOSUB32 is called, a new random number is formed
IX numeric mode flag. If IX is set to true before GOSUB32, only digits, decimal points and leading minus sign can be entered
OP number of options in a menu
Q$ leading question for INPUT (GOSUB32) or Y/N handler (GOSUB71)
Q$(1) to Q$(OP) options for MENU (GOSUB76)
YES true if answer to Y/N is YES, or number of selected option from MENU

Program Description
0-27: initialising, titling and rerouting.

Amend 26 if BACK-END is deleted.

32-68: INPUT subroutine

32-34: sets default length if necessary, moves cursor down, or scrolls. IR$, the right-of-cursor string is given a leading space which will be dropped later on. IV is vertical tab, and IB a local variable.

36-37: sets input string I$ and input character IC$ at zero length, prints the prompt question Q$ and the editable input heading on the top of the screen. It then PEEKs IH and IV, the horizontal and vertical tabs of the input character IC$. FOR IA = 0 TO 1 commences a truth loop which will repeat until a RETURN or ctrl-P is entered and IA set to 1 at line 49. Tabs are reset, I$ is extended by the latest input character, and the current input string length IM determined.

A character is lopped off IR$. When ID is true (delete mode flag), a character has been deleted so IR$ is reprinted one character to the left. When II is true (insert mode) IR$ is reprinted one character to the right.
37: first POKEs the keyboard stack clear then GETs a character IC$ from the keyboard. Its ASC equivalent is PEEKED rather than ASCed and stored in variable IC. ID the delete flag is turned off. If a ctrl character is pressed, the current character IC$ is made nil, II the insert mode flag is turned off and the program branches.
38: eliminates non-numerics in numeric mode. Minus sign is only possible in first character.
39-42: provides space for other input controls, for example, "macro"s, or setting flags for detecting the first occurance of some letter.
43-44: gives warning and prevents inputting too long a string.
46: prints the input character IC$ and increments the input-character tab IH. Unless a RETURN is detected, the program loops back to 35. Otherwise Q$, IX, IR$ and IL are reset, string-garbage FREed and all display right of the cursor deleted from the screen. IR is set by PEEKing at positions 78 and 79 which increment jat mega-speed during GET-waiting time, before RETURNing to the main program.
48: on ctrl-P I$ is set to include all the input, and IH set so that input right of the cursor is not wiped off.
49: on RETURN or ctrl-P, IC$ is set to chr$(13) and IA set to true to break the truth-loop.
50: on ctrl-B, input-character tab IH is reset to the beginning of the string, and I$ and IR$ adjusted to match. One blank space is added to the front of IR$, which gets knocked off next time round at line 42-43.
51: on ctrl-E, input character tab IH is set to the end of the right-of-cursor string IR$, and strings are amended to suit.
52-54: ctrl-H or back-space requires three lines, to prevent errant string commands. They handle the situations where I$ has length 0, 1, or more characters.
55-56: on ctrl-V or forward-space, the first character of IR$ is added to I$ (unless IR$ is a nil-string), and lopped off IR$ at line 36 or 37 next time round.
57: on cntl-C the Y/N subroutine is called after Bell. Responding Yes to the question will STOP the program.
58: on either ctrl-C or ctrl-I IH and IR$ are reset and the insert-mode flag set if appropriate.
59-60: on ESC, explanation of control characters is printed before IH and IR$ are reset.
61: on ctrl-D the delete flag is set.
63-67: lines free for adding further controls.
71-72: Yes/No subroutine. A truth-loop returns the program control to the beginning of line 71 in INVERSE mode if neither Y nor N has been pressed. Line 72 restores NORMAL and RETURNs with YES true or false.
76-77: MENU-GENERATOR subroutine. Clears the screen and displays the menu stored in Q$(1) to Q$(OP). A truth-loop assures a valid response or prints a warning. Asterisks are set by revising HTAB to the appropriate line, before HTAB returns below the display.
80-96: initiating subroutine. Add to it as appropriate.
1000-62999: demonstration program only. Replace with your own program.
63000-63700: reserved for error control. 63700 directs control to the lines which list an errant line.
63810: defines the BACK-END menu. The MENU subroutine returns the selected value of YES.
63900: hex-to-dec one-liner. One truth-loop will keep repeating prompts for hex numbers until a zero

is input, and another loop lops one character off the hex string until it is zero length, and increments C the dec value.
63910: dec-to-hex one-liner. Note the "0" is added to prevent bombing on a nil return.
63920-63922: amend printer control to suit your own machine. EXEC LISZT is Anderson et al's program cited below.
63930-63934: SAVE routine
63930: note the REM statement with 17 blank spaces in quotes into which the program name will be POKEd at 63934, and PEEKed at 63931 if no name is input.
63931: D is the current line number location, less 18 bytes, ie at the first space after the first quote in the REM statement in line 63930. If no name has been input, the name is PEEKed from the REM.
63932: A truth-loop insists on a legitimate drive number, or a RETURN-press. If a drive-number is given, it is added to H1$, the disk-command string.
63933: prints the disk-command after the OK is given to SAVE or CATALOG. Note use of the truth-loop and Y/N subroutine.
63934: the new program name is POKEd in the REM, with the first letter incremented.
63997-63999: Errors are sent to 63998, which determines the address of the 00000 in line 63997, and the line-number of the error. The last line POKEs this line-number into the 00000, clears the screen, POKEs 33,33 to remove unwanted blanks from the listing, and POKEs 216,0 to enable normal error diagnosis, then GOTOs 63997 to list the errant line and print the error.

REFERENCES
Anderson, Leonard H; Donald Cohen and Richard F Searle
LISZT with Strings
MICRO – the 6502/6809 Journal 48, May 1982,
pp37 – 45
Cheeseman, W J
INPUT$: A Friendly Substitute for INPUT
NIBBLE vol 3 no 4 1982
pp 115 ff
Glenn, Chris
GO – A Greeting Program
NIBBLE vol 3 no 4 1982
pp 109-113

```
SELECT FROM THE FOLLOWING:
  1...RUN
  2...HEX TO DEC CONVERSION
  3...DEC TO HEX CONVERSION
  4...KEY TO ASC
  5...PRINT LISTING
**6...SAVE
  7...RUN HELLO
  8...STOP
```

FIG 1 BACK-END MENU SHOWING CHOICE OF SAVE OPTION

```
RT-ARROW  RIGHT 1 SPACE
LF-ARROW  LEFT 1 SPACE
CNTL-B    TO BEGINNING
CNTL-C    STOP PROGRAM
CNTL-D    DELETE CHAR
CNTL-E    TO END OF INPUT
CNTL-I    INSERT MODE
CNTL-P    CNTL-E <RETURN>
```

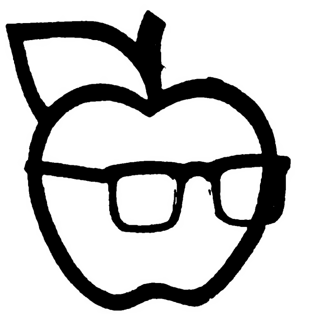
FIG 2 INPUT SCREEN WHEN <ESC> IS PRESSED

```
                                                    ENDS.NOREMS
                                                    OWEN FODGER
                                                     APRIL 1984
                                                       PAGE 1
22    GOSUB 82
     ONERR  GOTO 63000
23    GOTO 63800
32   IR$ = B1$+IR$
     ID = A1
     IL = IL+C1*(IL = A0 OR (IL+LEN(Q$)>C1))
     IV = B3
     #IF PEEK(B7)<IV THEN
          #  35
33   IV = B2-INT((IL+LEN(Q$))/B4)
    ┌FOR IB = A0 TO A1
    │    IB = (IV>PEEK(B7))
    │    #IF NOT IB THEN
    │         #  CALL -912
    │         #  POKE B7,PEEK(B7)-A1
34  └NEXT
     IV = PEEK(B7)
35   IC$ = B0$
     I$ = B0$
     VTAB IV+A1
     PRINT Q$;
     IH = PEEK(B6)+A1
     IV = PEEK(B7)+A1
     VTAB A1
     HTAB A1
     INVERSE
```

```
       PRINT "EDITABLE INPUT ON ---  <ESC> FOR HELP  "
       NORMAL
      ┌FOR IA = A0 TO A1
      │    IA = A0
      │    VTAB IV
      │    HTAB IH
      │    I$ = I$+IC$
      │    IM = LEN(I$)
      │    #IF LEN(IR$)<A2 THEN
      │         #  IR$ = B0$
      │         #   GOTO 37
 36   │    IR$ = MID$(IR$,A2)
      │    #IF ID OR II THEN
      │         #  PRINT IR$B1$
      │         #  VTAB IV
      │         #  HTAB IH
 37   │    POKE C2,A0
      │    GET IC$
      │    IC = PEEK(C3)
      │    ID = A0
      │    #IF IC<B5 THEN
      │         #  IC$ = B0$
      │         #  II = A0
      │         #  ON IC GOTO_

[37]  │          #  37,50,57,61,51,37,37,52,58,37,37,37,
                    49,37,37,48,37,37,37,
                 #  37,55,37,37,37,37,37,59,37,37,37,37
 38   │    #IF IX THEN
      │         #  #IF (IC<B1 OR IC>C0) AND NOT (IC=C4 AND IM=A∅) THEN
      │              #  PRINT DO$;

      │              #   GOTO 37
 43   │    #IF IM = (IL-A5) AND NOT IX THEN
      │         #  PRINT DO$;

 44   │    #IF IM = IL THEN
      │         #  IB = PEEK(49200)
      │         #   GOTO 37
 45   │    #IF II THEN
      │         #  IR$ = B1$+IR$
      │         #  #IF LEN(IR$)+IM = IL THEN
      │              #  PRINT DO$
      │              #   GOTO 36
 46   │    PRINT IC$
      │    IH = IH+A1
      └NEXT
      Q$ = B0$
      IR$ = B0$
      IX = A0
      IA = FRE(A0)
      IL = C1
      I = VAL(I$)
      IR = PEEK(B8+A1)*C1+PEEK(B8)
      VTAB IV
•     HTAB IH-A1
      CALL -958
      PRINT
      RETURN---->
48    I$ = I$+IR$
      IH = IH+LEN(IR$)
49    IC$ = CHR$(13)
      IA = A1
       GOTO 46
50    IH = IH-IM-A1
      IR$ = B1$+I$+IR$
      I$ = BO$
       GOTO 46
51    IH = IH+LEN(IR$)-A1
      I$ = I$+IR$
      IR$ = BO$
       GOTO 46
52    #IF IM THEN
           #  IR$ = MID$(I$,IM)+IR$
           #  IH = IH-A1
           #  #IF IM>A1 THEN
                #  I$ = LEFT$(I$,IM-A1)
53    IR$ = B1$+IR$
      #IF IM<A2 THEN
           #  I$ = BO$
54    IH = IH-A1
       GOTO 46
55    #IF LEN(IR$) THEN
           #  I$ = I$+LEFT$(IR$,A1)
           #  IH = IH+A1
56    IH = IH-A1
       GOTO 46
57    PRINT DO$
      HTAB A1
      VTAB A2
      INVERSE
      Q$ = "STOP PROGRAM"
       GOSUB 71
      #IF YES THEN
           #  HTAB A1
           #  VTAB IV+A1
           #  STOP
58    IH = IH-A1
      IR$ = B1$+IR$
      II = (IC = 9)
       GOTO 46
59    HTAB A1
      VTAB A3
      INVERSE
      PRINT "RT-ARROW  RIGHT 1 SPACE   "
      PRINT "LF-ARROW  LEFT 1 SPACE    "
      PRINT "CNTL-B    TO BEGINNING    "
      PRINT "CNTL-C    STOP PROGRAM    "
      PRINT "CNTL-D    DELETE CHAR     "
      PRINT "CNTL-E    TO END OF INPUT"
60    PRINT "CNTL-I    INSERT MODE     "
      PRINT "CNTL-P    CNTL-E <RETURN>"
      NORMAL
      IH = IH-A1
      IR$ = B1$+IR$
       GOTO 46
61    IH = IH-A1
      ID = A1 : I$=B1$
       GOTO 46
71   ┌FOR IB = A0 TO A1
     │   PRINT Q$"  Y/N ?  ";
     │   GET IH$
     │   IB = (IH$ = "Y") OR (IH$ = "N")
     │   #IF NOT IB THEN
     │        #  HTAB A1
     │        #  INVERSE
72   └NEXT
      NORMAL
[72]  PRINT IH$
      YES = (IH$ = "Y")
      Q$ = ""
      RETURN---->
76    HOME
      VTAB A4
      PRINT "SELECT FROM THE FOLLOWING:"
      VTAB A6
     ┌FOR IB = A1 TO OP
     │   HTAB A3
     │   PRINT IB"..."Q$(IB)
     └NEXT
     ┌FOR IB = A0 TO A1
     │   GET IH$
     │   YES = VAL(IH$)
     │   IB = (YES>A0) AND (YES< = OP)
     │   #IF NOT IB THEN
     │      #  PRINT "OUT OF RANGE, TRY AGAIN"
77   └NEXT
      VTAB (A5+YES)
      PRINT "**"
      IA = FRE(0)
```

```
        VTAB A8+OP
        RETURN---->
82      A0 = 0
        A1 = 1
        A2 = 2
        A3 = 3
        A4 = 4
        A5 = 5
        A6 = 6
        A7 = 7
        A8 = 8
        A9 = 9
83      B0 = 16
        B1 = 46
        B2 = 22
        B3 = 13
        B4 = 40
        B5 = 32
        B6 = 36
        B7 = 37
        B8 = 78
84      C0 = 57
        C1 = 256
        C2 = -16368   ── C4=45
        C3 = -16384
86      D$ = CHR$([illegible])+CHR$(A4)
        DO$ = CHR$([illegible]7)
87      BO$   ""
        B1$   " "

[87]    BL$ = "
96      RETURN---->

1000        REM
63000       REM

63700    GOTO 63998
63800   OP = A8
        Q$(A1) = "RUN"
        Q$(A2) = "HEX TO DEC CONVERSION"
        Q$(A3) = "DEC TO HEX CONVERSION"
        Q$(A4) = "KEY TO ASC"
        Q$(A5) = "PRINT LISTING"
        Q$(A6) = "SAVE"
        Q$(A7) = "RUN HELLO"
        Q$(A8) = "STOP"
63812    GOSUB 76
        ON YES GOTO 1000,63900,63910,
        63950,63920,63930,63940,63960
63900  ┌FOR D1 = A0 TO A1
       │   Q$ = "HEX "
       │   IL = 10
       │    GOSUB 32
       │   I$ = "0"+I$
       │   D1 = (I$ = "0")
       │   C = A0
       │  ┌FOR D = A0 TO A1
       │  │   C = C*B0+ASC(I$)-48-A7*(I$>"9")
       │  │   I$ = MID$(I$,A2)
       │  │   D = (LEN(I$) = A0)
       │  └NEXT
       │   PRINT "DEC="C
       └NEXT
        GOTO 63800

63910  ┌FOR D1 = A0 TO A1
       │   Q$ = "DEC "
       │   IL = 10
       │   IX = A1
       │    GOSUB 32
       │   D1 = (I = A0)
       │   H$ = ""
       │  ┌FOR D = A0 TO A1
       │  │   X = I-INT(I/B0)*B0
       │  │   H$ = CHR$(X+48+A7*(X>A9))+H$
       │  │   I = INT(I/B0)
       │  │   D = (I = A0)
       │  └NEXT
       │   PRINT "HEX = "H$
       └NEXT
        GOTO 63800
63920   OP = A2
        Q$(A1) = "EXEC LISZT"
[63920] Q$(A2) = "COMPACT LISTING"
        GOSUB 76
        #IF YES = A1 THEN
           #  PRINT D$"EXECLISZT"
63922   PR# 1
        POKE 1657,75
        LIST
        POKE 1657,40
        PR# 0
        STOP
```

```
63930  ┌FOR YES = A0 TO A1
       │  Q$ = "NAME PROGRAM OR TYPE CATALOG:"
       │  IL = B0
       │   GOSUB 32
       │  #IF I$ = "CATALOG" THEN
       │      #  H1$ = D$+I$
       │         #  REM  "                        "

63931     D1 = PEEK(121)+PEEK(122)*C1-B0-A2
          #IF I$<>"CATALOG" THEN
             #  I$ = "A"+I$
             #  H1$ = D$+"SAVE"+I$
             #  #IF I$ = "A" THEN
             #   I$ = ""
             #  ┌FOR C = A0 TO B0
             #  │#  I$ = I$+CHR$(PEEK(D1+C))
             #  └NEXT
             #  H1$ = LEFT$(H1$,LEN(H1$)-A1)+I$
63932    ┌FOR C = A0 TO A1
         │  PRINT "DRIVE NUMBER?"
         │  GET H$
         │  C = (H$ = CHR$(B3)) OR H$ = "1" OR H$ = "2
         └NEXT
          #IF H$<>CHR$(B3) THEN
             #  H1$ = H1$+",D"+H$
63933     Q$ = "CONFIRM..."+MID$(H1$,A3)
           GOSUB 71
       └NEXT
       PRINT H1$
       #IF I$ = "CATALOG" THEN
          #  PRINT "PRESS A KEY TO CONTINUE"
          #  GET H$
          #   GOTO 63800
          63934   I$ = LEFT$(CHR$(ASC(LEFT$(I$,A1))+A1)
                         +MID$(I$,A2)+BL$,B0+A1)

                 ┌FOR C = A0 TO B0
                 │   POKE D1+C,ASC(MID$(I$,C+A1,A1
                 └NEXT
                  GOTO 63800
         63940   PRINT D$" RUN HELLO"
         63950  ┌FOR D = A0 TO A1
                │   PRINT "PRESS A KEY "
        [63950] │   GET H$
                │   PRINT H$"  "ASC(H$)
                │   D = (ASC(H$) = B3)
                └NEXT
                ┌FOR D = A0 TO 1000
                └NEXT
                 GOTO 63800
         63960   STOP
                 GOTO 63800
         63997   LIST 00000
                 RESUME
         63998   D = PEEK(121)+PEEK(122)*C1-A7
                 C = PEEK(218)+PEEK(219)*C1
                 H$ = RIGHT$("0000"+STR$(C),A5)
         63999  ┌FOR C = A0 TO A4
                │   POKE D+C,ASC(MID$(H$,C+A1,A1)
                └NEXT
                 TEXT
                 POKE 33,33
                 POKE 216,0
                  GOTO 63997


                 END OF LISTING


        PROGRAM LENGTH = 3467 BYTES
        TOTAL OF 57 LINES

        294 TOTAL NON-REM STATEMENTS,  3 TOTAL REM

        END
```



THE WORM IN THE APPLE

How well will they work?

Strange how we worms get to hear things that don't reach human ears. Here we have all this wondrous publicity for the new mini-floppy disks (which I think should be called micro-semi-hard but no one is listening). These disks will sweep all before them, they will transform the industry. In two years, we are assured, they will be responsible for 60% of the disk market. For all I know, they will leap tall buildings in a single bound.

But no one has asked the simple question:

How well will they work?

In the handbook for the Hewlett Packard 150 computer (a very pleasant machine suffering from a small screen) there are a series of chilling paragraphs that suggest the floppies have a finite life.

That after "x" amount of use they can no longer record data.

As far as this worm has been able to find out, no other manufacturer has stood up bravely and said these disks cease to work after you have used them for a period of time.

Is it that Hewlett Packard knows something Apple doesn't.

Surely not.

Down the gurgle

My transatlantic cousin tells me that Franklin Computers has gone down the gurgle.

If you recall, they made an Apple clone which they stated could not possibly be considered an imitation because they had consulted with higher authorities (presumably not of an earthly kind) and been given a celestial OK. This worm, in its own crawling way, quietly didn't believe them.

And sure enough they had their day in court. Now they have disappeared from our ken.

This belief that an Apple clone cannot be successfully sued in court is quite widespread. I'm sure I heard somebody saying something of the sort only the other day.

Died the death

As I am only a lowly worm and cannot expect to be kept informed on these matters I am still not certain whether the Apple III has ceased production, although I am told such statements have appeared in the newspapers so they must be true.

It is always a pity to see a computer disappear.

True it was born in confusion and ineptly made but the bugs were quickly sorted out and it became, in this worm's humble opinion, an excellent business machine.

I asked the esteemed company of Burson Marsteller whether the Apple III had died the death. They knew not. Perhaps Apple hadn't told them.

My transatlantic cousin tells me that the cause for its untimely demise was partly that it had run foul of the Federal government by not meeting their radio frequency interference standards and that an Apple III Plus (there is true originality for you) had actually been released in the United States – it never, as far as this worm knows, arrived in Australia. The new machine had a four layer motherboard and complete conformity to the government's standards.

It also had as extra add-ons, a clock with calendar, a keyboard copied from the IIe, a much more powerful power supply and far better graphics.

All these features could be retrofitted to the original machine. But it still didn't sell.

In the end, Apple probably sold more than 80,000 and less than 100 ,000 Apple IIIs, which for a company their size is a worm hole in the snow in sales.

My transatlantic cousin has further told me that at least five companies have approached Apple for permission to carry on manufacturing the Apple III under another name.

One of the companies who made such an approach was the ill-fated Franklin. They suggested they relaunch the machine under the quite strange name of "The Enforcer".

No one knows how they came up with that idea.

Anyway the answer was no and the Apple III, so my transatlantic cousin tells me, is definitely dead. No flowers by request.

Scurrilous rumour

This magazine has no place for rumour, scurrilous or otherwise, and therefore the story that Apple are about to release two new versions of the Apple II are totally and utterly untrue and will be ignored.

Equally untrue is that one of them will be a severely stripped down version of the IIe selling for very small money indeed. Figures around the magical $500 have been mentioned.

Even more untrue is that an enhanced IIe with detachable keyboard, two built-in disk drives, a numerical keypad and user definable keys will be released.

Where, I often wonder, do these lying rumours start? Certainly not in this magazine.

Apple inundated?

Apple made an offer of new lamps for old in Melbourne. They offered to take fake Apples in part exchange for new Apples.

Was this scheme a success?

Was Apple inundated with fake Apples?

And if so, why don't Apple proudly proclaim their achievement?

And if not why don't they tell us that the fake Apple population of Victoria is smaller than anyone thinks? As a worm it is hardly my place to press for an explanation of this strange series of events.

Gross carelessness

My English cousin tells me that to an outsider it has been the night of the long knives in English Apple headquarters in the Old Dart. No less than six senior employees left simultaneously.

Apple in England stoutly deny there has been a purge. And we believe them.

Still, to lose your sales and marketing director, your product marketing director, your dealer sales manager, your vertical markets manager, one of your technical support people and a senior executive from the Lisa division all at once seems a little odd.

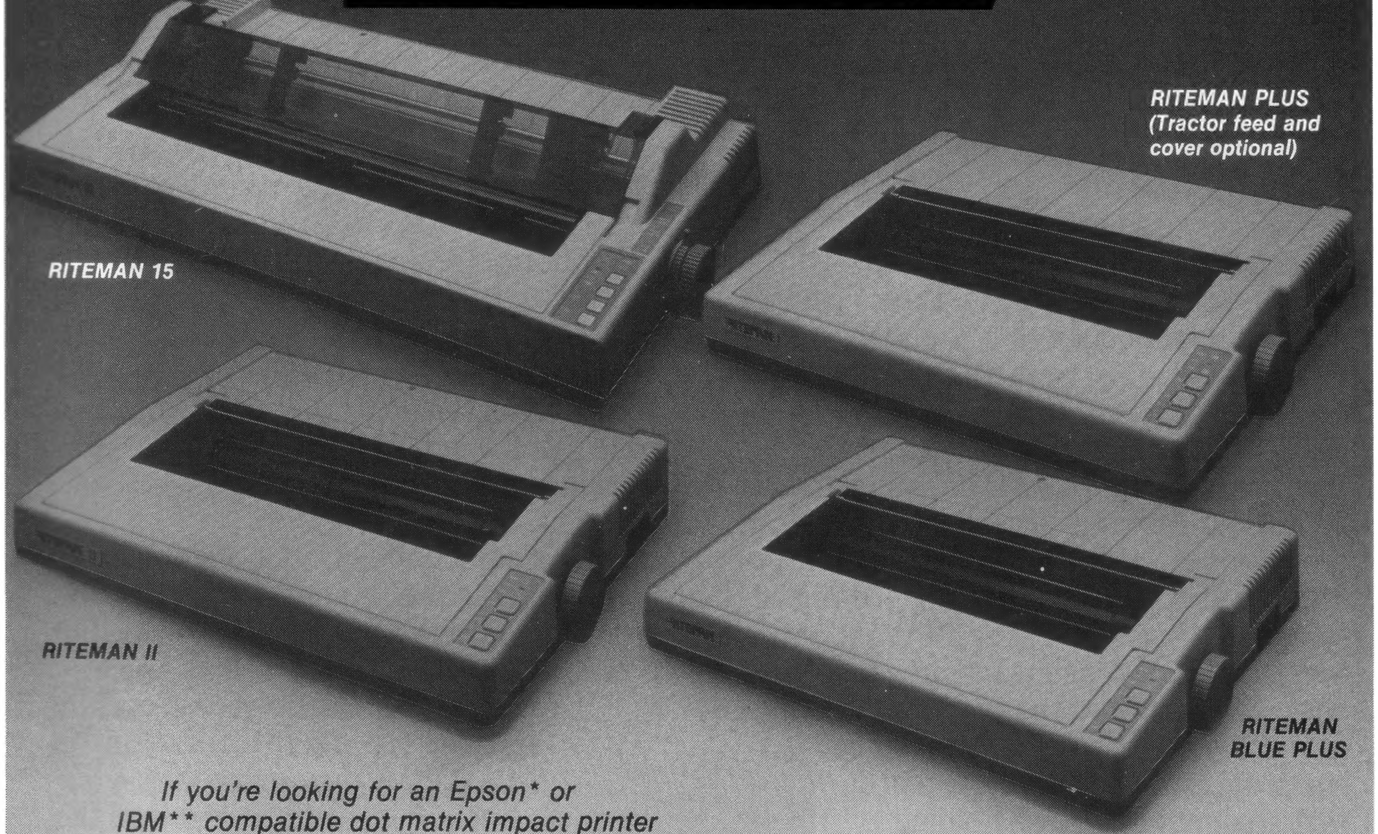
To misquote Lady Bracknell in "The Importance of Being Earnest", "to lose one executive may be considered unfortunate, to lose both is gross carelessness".

Quite so.

What does that make losing six at once? ■

RIGHT PRICE. RIGHT SIZE. RIGHT SPEED.

That's Riteman!



If you're looking for an Epson or IBM** compatible dot matrix impact printer that's lower in price, smaller, durable, and fast enough to keep pace with your work flow . . . we'd like to show you a Riteman!*

Four great new printers. All priced well against comparable Epson models. All substantially smaller. Yet solid as a rock. With print speeds up to 50% faster than the competitors.

The Riteman Plus, Riteman II, Riteman 15, and Riteman Blue Plus with IBM graphics. All are backed by a full one-year warranty. And, as you can see, they're great-looking, too!

So see your dealer today . . . and get the Riteman for your printing job!

* Epson is a registered trademark of Epson America, Inc.
**IBM is a registered trademark of International Business Machines Corp.

RITEMAN

For further information contact your local Warburton Franki office.

WF Warburton Franki

A DIVISION OF THE ANI CORPORATION LTD (INCORPORATED IN N.S.W.)

● ADELAIDE (08) 356-7333 ● BRISBANE (07) 277-0222 ● HOBART (002) 28-0321
● MELBOURNE (03) 795-9011 ● PERTH (09) 277-7000 ● SYDNEY (02) 648-1711
● AUCKLAND N.Z. (09) 50-4458 ● WELLINGTON N.Z. (04) 69-3016

WF934/81

Introducing a Teacher You Can Give to Your Apple.

For the first time in Australia, quality educational courseware is available for your Apple IIE or your Apple II Plus. Control Data PLATO® is proud to announce that from December 1st, new lessons from the vast PLATO computer based education library will be available for your Apple micro.

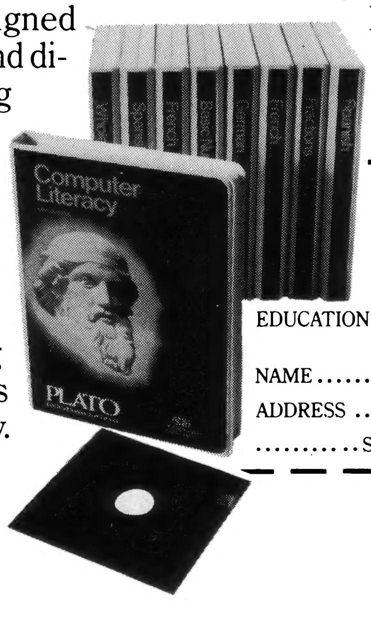
Control Data PLATO computer-based education, recognised as a world leader in educational software, is a highly interactive system of self-paced, one-to-one instruction. Developed over 20 years, it has been thoroughly tested and proven in business, industry, and the academic community. PLATO lessons are designed and programmed to motivate and direct students as well as helping them develop a sense of accomplishment and personal progress.

For example, the course titled Computer Literacy provides the foundation for simple programming as well as giving a brief introduction to the uses of computers in today's society. There are lessons in building vocabulary in French, German and Spanish as well as lessons in Physics, Fractions, Decimals and Basic Number Facts.

Today, more than ever before, all members of society need to learn. PLATO computer-based education is designed and paced for the individual. This means easier learning with infinite patience and personalization.

As traditional education methods become progressively more costly and less efficient you owe it to yourself to bring the power of PLATO learning to your Apple.

If you would like to be on our mailing list for future information on PLATO Courseware for your Apple, please fill in and mail the coupon below.



PLATO

Changing How The World Learns.

Mail to: Marketing Manager,
EDUCATION SERVICES, CONTROL DATA AUSTRALIA PTY. LTD.,
493 St. Kilda Road, Melbourne 3004.

NAME ....

ADDRESS ....

.... STATE .... POSTCODE ....

CONTROL DATA

Burrows Doble Lawrence CDA/616